Carnegie Mellon University
The Robotics Institute

Working Papers **Journal**

Volume 8 | Fall 2020

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

Carnegie Mellon Robotics Institute Summer Scholars

Working Papers **Journal**

RISS Director

Dr. John M. Dolan

jdolan@andrew.cmu.edu

RISS Co-director

Ms. Rachel Burcin

rachel@cmu.edu

2020 RISS Journal Team

Josiah Coad
Hunter Damron
Ethan Fahnestock
Shivesh Khaitan
Rebecca Martin
Willa Potosnak
Nayana Suvarna
Renos Zabounidis

We gratefully acknowledge the support of the National Science Foundation through the Research Experience for Undergraduates (REU) program (Grant # 1659774).

The Robotics Institute Summer Scholars Working Papers Journal is an annual publication of the Robotics Institute's Summer Scholars Program at Carnegie Mellon University.

Copyright © Carnegie Mellon University 2020.

Table of Contents

Table of Contents

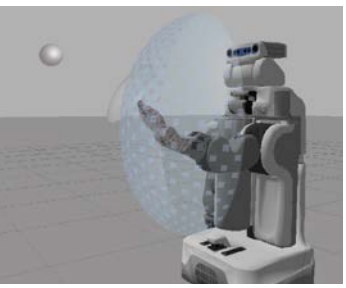
Continued

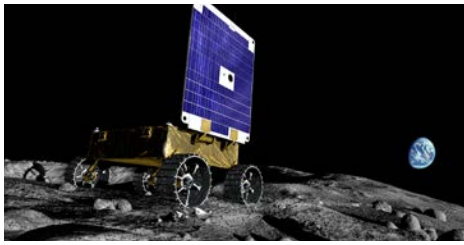
* *this paper was authored by three scholars, Rebecca Martin, Lan Wu, and Yu Wu*

Introduction

The Robotics Institute Summer Scholars Working Papers Journal is an annual publication of the Robotics Institute's Summer Scholars Program at Carnegie Mellon University. The journal is a medium for the undergraduate students of the summer research program to communicate their work in collaboration with the participating lab faculties. This journal encompasses the learnings and research findings of the students over the eleven-week long remote-engagement with the CMU community for the year 2020.

The journal comprises 40 papers written by the scholars participating in RISS 2020. The papers included explore varied domains of Robotics including: Localization, Mapping, Computer Vision, Motion-planning, Controls, Aerial Systems, Multi-agent Systems, Machine Learning, and Reinforcement Learning.





The papers have been drafted by the scholars in collaboration with graduate student and faculty mentors. The scholars would like to thank all the mentors for their invaluable guidance and feedback throughout the program. Their expertise has been of immense support for the scholars.

The scholars would also like to acknowledge the support provided by the Global Communication Center (GCC). The GCC held several workshops and one-on-one appointments for individual scholars throughout the program. Their assistance in guiding and reviewing the individual works has helped the scholars to acquire necessary skills for writing and presenting their work. We are grateful to Professor Joanna Wolfe and the whole GCC team for their support.

Finally the cohort would like to thank RISS co-directors Rachel Burcin and Dr. John M Dolan who have put their efforts in making this program possible even in the most difficult time of global pandemic. Their ability to quickly adapt the program and welcome each scholar to the CMU community while overcoming the barriers introduced by the virtualization of the program was inspiring, and the RISS experience would not have been possible without their hard work and coordination. We are also thankful to the whole CMU community for their contribution to the program.

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

Dear Colleagues

The Carnegie Mellon Robotics Institute is committed to opening doors and creating opportunities for future leaders in robotics. The RI Summer Scholar's Program (RISS) is a high impact student development program with a distinctive community & culture that is inclusive and accessible. Through RISS, CMU welcomes students from across the country and world opportunities to conduct research with robotics researchers at the Robotics Institute. The quality and breadth of research, high level of institute and university engagement, and powerful professional development programming, graduate school application counseling, and alumni network create transformative experiences and remarkable post-program trajectories.

The RISS program immerses students in the world of robotics. The unique post-program mentoring and coaching provides spectacular results with more students from this program than any other at CMU (that we know of) successfully awarded admissions to multiple programs across the School of Computer Science. The scholars are innovative and talented emerging scholars.

This year 2020 was a true challenge. But despite our borders being closed amid COVID-19, thanks to the support of so many, the RISS community contributions and partnership were able to create possibilities for undergraduate scholars.

With 43 scholars from 11 countries and 33 universities, the 2020 RISS cohort was the largest cohort since the inception of the program in 2006. Over forty percent of our participating scholars are from communities underrepresented in STEM. Twenty-four of the 2020 scholars are from the United States, a significant point that showcases a push in our country to support STEM research and education. The RISS 2020 international scholar community is represented by 19 scholars from 10 different countries.

The RI scholars' research experience is guided by outstanding research mentors that represent the incredibly diverse robotics research from across the Institute. This year 25 individuals and their teams undertook the challenge of navigating a remote experience. The RISS community welcomed eight new mentors and thanked the many returning mentors who have had a tremendous impact on the lives and careers of the scholars. Mentors guide, support, create new possibilities, and nurture students' potential.

With gratitude,

John & Rachel

Dr. John M. Dolan
Director of RISS Program
& Principal Systems Scientist
jdolan@andrew.cmu.edu

Ms. Rachel Burcin
Co-Director of RISS Program &
Global Programs Manager
rachel@cmu.edu

Thank You

Program Sponsors & Partners

As all aspects of the RISS experience moved online, the scholars of 2020 and the entire RISS community were especially fortunate to have a community of advocates standing with them. We extend thanks to all who supported this opportunity, including the 25 RISS admissions committee members who helped to select the 2020 cohort; the RI graduate students and staff mentoring and guiding; presenters sharing their expertise and journeys in STEM; and the student affairs, university, and professional development leaders providing online professional development.

NSF

Carnegie Mellon University
The Robotics Institute

Carnegie Mellon University

Carnegie Mellon University
School of Computer Science

Carnegie Mellon University
Mechanical Engineering

Carnegie Mellon University
LEONARD GELFAND CENTER
FOR SERVICE LEARNING AND OUTREACH

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

open robotics

UNITED STATES
AIR FORCE ACADEMY

KGSP
برنامج جامعة الملك عبدالله للعلوم والتقنية للطلبة الموهوبين
KAUST GIFTED STUDENT PROGRAM

上海科技大学
ShanghaiTech University

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

UBTECH

NREC
EXPERIENCE · EXPERTISE · EXCELLENCE

Mobility21
A USDOT NATIONAL UNIVERSITY TRANSPORTATION CENTER

CMU Global Communication Center

Thank you to the incredible community who generously gave their time & expertise to guide and mentor the 2020 RI Summer Scholars.

The scholars would like to especially thank all those who helped to shape the work contained in this journal. We thank those listed below and the many more that have invested in the futures of these students.

Aaron M. Johnson
Abhinav Gupta
Achal Dave
Ada Taylor
Andrew P. Sabelhaus
Andrew Saba
Anthony Wertz
Arman Kilic
Artur Dubrawski
Benedikt Boecking
Carmel Majidi
Chen Fu
Chen Wang
Christoph Mertz
Dana Hughes
Dana Hughes
David Held
Deva Ramanan
Ding Zhao
Fan Jia
Guillaume Sartoretti
Haidar Jamal
Heather Jones
Henny Admoni
Hernando Gomez
Howie Choset
Huao Li
Isaac Isukapati
Ishani Chatterjee
Jack Mostow
James K. Miller
Joanna Wolfe
John M. Dolan
Joshua Spisak
Julian Whitman
Juliann Reineke
Katia Sycara
Keith A. Dufendach
Maggie Collier
Maxim Likhachev
Mengdi Xu
Michael Pinsky
Mohammadreza Mousaei
Nirmal Patel
Oliver Kroemer
Qin Lin
Raewyn Duvall
Ramkumar Natarajan
Reid Simmons
Reuben Aronson
Robert Edman
Rogerio Bonatti
Roshni Kaushik
Sam Powers
Sean J. Wang
Sebastian Scherer
Shivam Vats
Shu Kong
Shuo Yang
Siddarth Agrawal
Skye Thompson
Stephen Smith
Tejus Gupta
Thomas Weng
Tianwei Ni
Tushar Kusnur
Victoria Dean
Vidhi Jain
Wenhao Luo
William Whittaker
Xinyu Li
Yikang Gui
Yuheng Qiu
Zhaoyuan Gu
Zhiqian Qiao

43 SCHOLARS
FROM **11** COUNTRIES
& **33** UNIVERSITIES

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

Comparing State Representations for Learning Deformable Object Manipulation Policies

Khush Agrawal[1], Thomas Weng[2], and David Held[2]

***Abstract*— Recent advances in object manipulation have been largely focused on rigid-body manipulation. For tasks like laundry folding and cable management, deformable properties of objects cannot be ignored. Rigid body assumptions can cause a system to fail in such scenarios. We propose a two-stage algorithm to handle a given deformable object manipulation task. The first stage represents a deformable object by the 3-D coordinates of the four corners of a fabric. The second stage learns to predict actions to manipulate a fabric. We compare our technique with a dense representation based technique and show that our method is able to learn lifting and folding tasks in 13 and 36 episodes respectively.**

***Index Terms*— Manipulation, Representation Learning, Reinforcement Learning**

I. INTRODUCTION

Robotic manipulation of deformable objects has applications in both household tasks like bed-making and folding laundry, as well as industrial tasks like cable management and textile manufacturing. Knowledge of deformable objects is vital to successfully complete such tasks. A large part of manipulation literature focuses on rigid-body manipulation [1], [2], often relying on assumptions that only hold true for rigid objects. These rigid body assumptions make a robot unable to handle tasks where deformations need to be considered. Selecting an appropriate representation for a deformable object is a key element in successfully completing a manipulation task. Estimating the states of a deformable object is a challenging task due to the infinite number of possible configurations. This difficulty in state estimation makes a manipulation task for deformable objects like fabric more challenging compared to their rigid object counterparts. In this paper, we decouple a manipulation task into two stages. The first stage represents a deformable object by the 3-D coordinates of its corners and the second stage learns a manipulation policy using this representation. We compare our approach, which is an oracle approach against the descriptor-based representation described in [3]–[5] and show that our 3-D coordinate based representation makes policy learning more sample efficient on the simulated folding and lifting tasks shown in Fig. 1.

[1]Khush Agrawal is with the Department of Electronics and Communication Engineering, Visvesvaraya National Institute of Technology, Nagpur, India and Robotics Institute Summer Scholars, Carnegie Mellon University, Pittsburgh, USA. khush@students.vnit.ac.in

[2]Thomas Weng and David Held are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA. {tweng, dheld}@andrew.cmu.edu

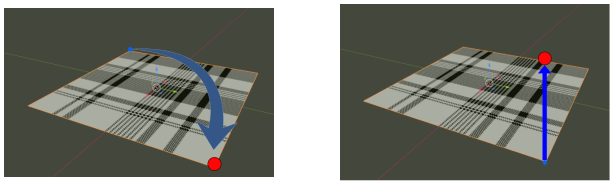


Fig. 1. Task definition figures. The goal of the folding task (left) is to perform a diagonal fold by bringing the manipulated cloth point (blue sphere) across the cloth to the goal point (red sphere). For the lifting task (right), the goal is to bring the manipulated cloth point to a goal pose above it.

II. RELATED WORK

We have divided the previous work into two parts: representation learning and policy learning for deformable object manipulation.

A. Representation Learning for Deformable Objects

Previous methods on learning a representation for a deformable object used approaches like modeling the entire cloth [6] using depth image segmentation and volumetric fusion. Some methods like [1], [2] compress high dimensional RGB images into a latent embedding which makes policy learning sample-efficient. This method performs reasonably on rigid body manipulation tasks like sliding a puck, opening a door, but has difficulty capturing fine details like the folds and wrinkles of a cloth, limiting their effectiveness for such deformable objects. Other methods use a subset of 3-D coordinates of points on a fabric [7]. This method successfully solves simple manipulation tasks like folding, placing on a table, etc. We build up our method on this approach and compare it with higher-dimensional representations. Approaches like [3]–[5] use high dimensional descriptor to represent a cloth. Although it has been demonstrated for tasks like manipulating a shoe, soft-toy, we experimentally show that it is unable to perform well on fabric manipulation tasks.

B. Policy Learning for Deformable Objects

Some methods like [8] develop a predictive model that learns the forward model of a fabric. The forward model predicts the states of a deformable object by taking an action and its previous state as an input. This can be further coupled with a model-based planning method like Model Predictive Control (MPC) to predict the optimum sequence of actions to manipulate an object to the goal state. However, the forward models can be extremely sensitive to deformation. Other methods like [7] train a model-free algorithm like Deep Deterministic Policy Gradient (DDPG) along with demonstrations and solve manipulation tasks in simulation.

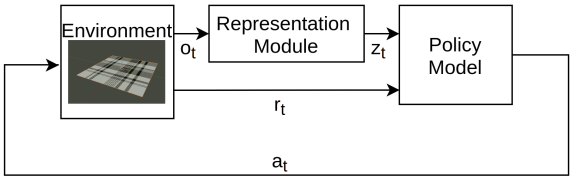


Fig. 2. Diagram for the two stage framework depicting the flow of observations, rewards and actions.

Additionally, the algorithm can solve dynamic manipulation tasks like placing a cloth on a table where the working space of a manipulator is constrained within certain limits. We compare the approaches proposed in [3]–[5], [7] and evaluate the results on folding and lifting tasks (Fig 1).

III. METHOD

We decouple the deformable object manipulation task into two sub-tasks: representation and policy learning (Fig. 2). The former model outputs a representation for fabric and the latter uses this representation to predict actions to manipulate the fabric to the goal state. The following subsections provide details about the two models.

A. Representation Model

We demonstrate the use of 3-D coordinate based representation for fabric and compare our method with a high-dimensional representation based method.

1) 3-D Coordinate Representation: We simulate the experiments in Blender [9], an open-source simulation and rendering engine, which internally represents fabrics as a 3-D polygon mesh. Blender's API provides access to the 3-D coordinates of the mesh vertices. We consider a subset of these 3-D coordinates of the mesh as a representation of the fabric as shown in Fig. 3. As proposed by [7], we concatenate the 3-D coordinates of the corners of the fabric and use it as a representation of the fabric. We experimentally found out that this approach is effective in solving simple tasks like folding and lifting.

B. Policy Model

1) Background: We frame the cloth manipulation problem as a Partially Observable Markov Decision Process where the environment is defined by a $(S, A, P, r, O, \rho_0, \gamma)$ transition tuple. S is a set of full states of the environment, A is a set of continuous actions, $P : S \times A \times S \rightarrow R$, is the transition probability distribution, $r : S \times A \rightarrow R$ is the reward function, ρ_0 is the initial state distribution, and $\gamma \in (0, 1]$ is the discount factor. The decision process is partially observable and the agent receives observations o (3-D coordinates of four corners) from the set of observations O (3-D coordinates of all the mesh vertices).

The goal of the agent is to maximize the multi-step return $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the fixed horizon for each episode. The objective during learning is to find an optimal policy $\pi^* : o \rightarrow A$.

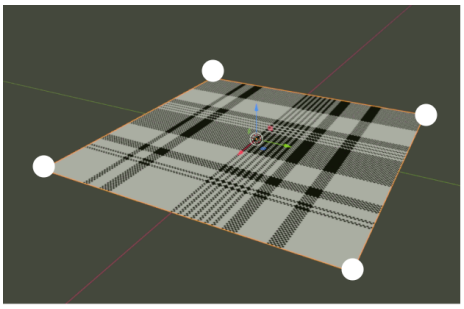


Fig. 3. Simulator snapshot. White spheres indicate the selected points for representing a fabric.

2) Policy Training: As shown in Fig. 2, we pass the observation (o_t) returned by the environment through the representation model which returns a representation (z_t). z_t is then passed through a policy network which predicts three Gaussian distributions $\mathcal{N}(\mu, \sigma^2)$ parameterized by their mean μ and standard deviation σ, corresponding to the displacements along $\vec{x}$, $\vec{y}$, $\vec{z}$ axes at time step t. The actions are as follows.

$$a_t = (\Delta x, \Delta y, \Delta z)$$

Δx, Δy, Δz are the displacements along $\vec{x}$, $\vec{y}$, $\vec{z}$ axes respectively. The transition tuples $(z_t, a_t, r_t, z_{t=1}, d)$ generated at every step are stored into a replay buffer R. At every step, transition tuples are randomly sampled from R.

Policy training is performed using the DDPG algorithm [10] with Adam [11] as the optimizer. We considered four networks: actor-target, actor-behavior, critic-target and critic-behaviour parameterized by $\theta^{\mu'}, \theta^{\mu}, \theta^{Q'}, \theta^{Q}$, respectively that are updated as follows. Critic-behavior parameters are updated to minimize the following loss.

$$L = \frac{1}{N} \sum_i (r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) - Q(s_i, a_i|\theta^{Q}))^2$$

Actor-behavior parameters are updated using the chain rule.

$$\nabla_{\theta^\mu} J(\theta) \approx \nabla_a Q(s, a) \nabla_{\theta^\mu} \mu(s|\theta^\mu)$$

Target network parameters are updated as follows.

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1 - \tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1 - \tau)\theta^{\mu'}$$

where $\tau << 1$.

IV. EXPERIMENTS

We compare the 3-D coordinate representation against a method that uses dense descriptors as the representation [3]. The dense descriptor network takes an image as input and outputs per-pixel latent descriptors. This representation must be learned, and we discuss the training procedure in the following subsection. As the 3-D coordinate based representation is an oracle representation, we expect it to outperform the dense descriptor based representation and the results align with our hypothesis.

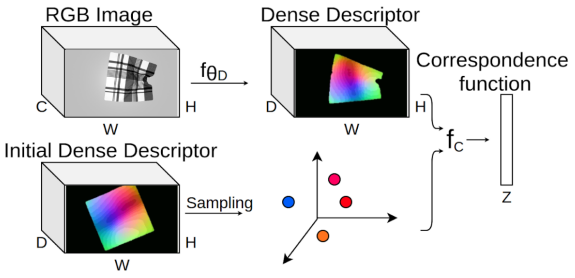


Fig. 4. Overview of extraction of sparse keypoints from a dense descriptor by comparing with an initial descriptor set.

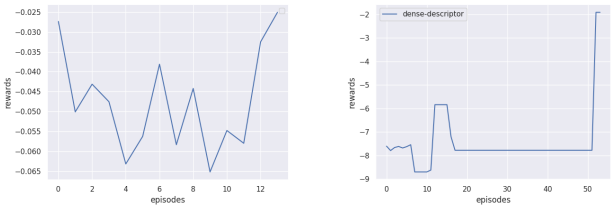


Fig. 5. Rewards collected over an episode. The policy network is able to learn the lifting tasks in 13 and 52 episodes for 3-D coordinate and dense descriptor based representations respectively. Left figure denotes rewards for 3-D coordinate based training. Right figure denotes rewards for dense descriptor based training.

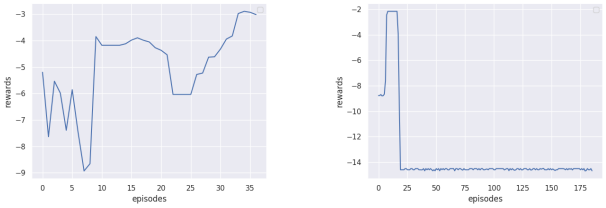


Fig. 6. Rewards collected over an episode. The policy network is able to learn the folding task in 36 episodes using 3-D coordinate based representation (left) whereas it fails for dense descriptor based representation (right).

A. Dense Descriptor Representation

We use the Blender simulator described previously to create a synthetic dataset for training the descriptors. We gather trajectories of random cloth movements, with the simulator providing a RGB image at every time step. We obtain the ground truth pixel correspondences between images of the cloth in different configurations directly from the simulator. Using this dataset of images and corresponding pixels, we train a 34-layer, stride-8 ResNet in Siamese fashion to bring corresponding pixels closer in descriptor space, while pushing non-corresponding pixels further away. As shown in Fig. 4, the $H \times W \times C$ RGB image is passed through a representation network f_{θ_D} which outputs a $H \times W \times D$ descriptor, where C denotes the number of image channels, and D is the length of the descriptors. Training a policy on this higher dimensional representation is challenging due to the curse of dimensionality, which motivates us to perform sparse keypoint extraction. We follow [4], where the dense descriptor output is compared with an initial sparse descriptor set at every time step using a correspondence function f_C to produce (u, v, z) representation of the fabric, where u, v are the image coordinates of the corresponding descriptor pixels and z is the depth coordinate. Results for the descriptor training can be seen in the Appendix.

B. Policy Learning

We create an OpenAI-Gym like environment using Blender to return a (observation, reward, done) tuple at every time step. The specifics of (observation, reward, done) are mentioned in the following section.

For evaluating our method, we consider the following tasks.

1) Lifting to location: As shown in Fig. 1, a cloth placed on a table is to be lifted to a goal location. The goal state is achieved if the L2 norm between the four corners of the present state and the goal state is less than a threshold distance ($\delta = 0.02$m). The horizon T is set to 20 steps. The results are visualized in Fig. 5.

2) Diagonal Folding: As shown in Fig. 1, a cloth placed on a table is to be folded diagonally to a specified location. The goal state is achieved if the L2 norm between the four corners of the present state and the goal state is less than a threshold distance ($\delta = 0.02$m). The horizon T is set to 40 steps. The results are visualized in Fig. 6.

We evaluate the performance with the following reward function.

1) L2 Norm: At every time step t, we provide the following reward.

$$r_t = -\left\|(c_t - g)\right\|$$

where c_t, g corresponds to the 3-D coordinates of the corners of the fabric at time step t and goal state respectively. We experimentally observed that the 3-D coordinate based representation perform better with squared L2 norm rewards.

V. DISCUSSION

We described an algorithm to learn a deformable object manipulation policy using a two-stage approach and demonstrated it on diagonal folding and lifting tasks. When dealing with tasks which involve multiple folds and highly crumpled configurations, however, we suspect that the proposed method can fail. Four corner coordinates for a deformable object are likely insufficient to describe a crumpled cloth. Further investigation is required to determine if a more robust method is necessary for more difficult tasks. As proposed in as [7], we plan to train a policy for more complex tasks like placing a cloth on a table, where the working space of a manipulator is constrained and it needs to learn to swing the cloth to place it on a table. This can also be potentially transferred to the real-world by learning a model for 3-D keypoint estimation in the real-world.

APPENDIX

This section consists of results from the training procedure of a 3 channel dense descriptor for fabric evaluated on a set

of 100 pairs of held-out set of validation images with 100 annotations in each pair.

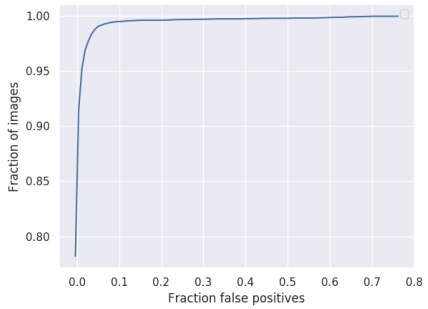


Fig. 7. CDF plot of the fraction of pixels in image pairs which are closer in descriptor space than the true match.

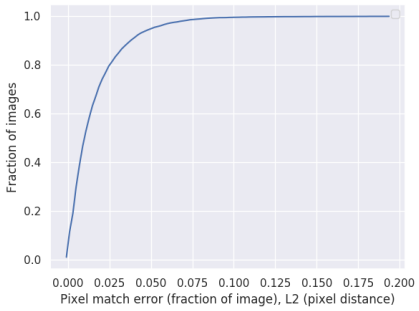


Fig. 8. CDF plot of the L2 distance between predicted match and ground truth label normalized by 800 (diagonal length for a 640 $\times$ 480 image).

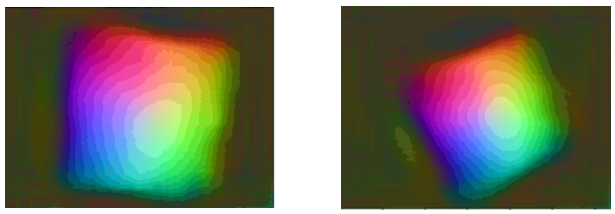


Fig. 9. Heatmap visualization of the three channel descriptors for a fabric.

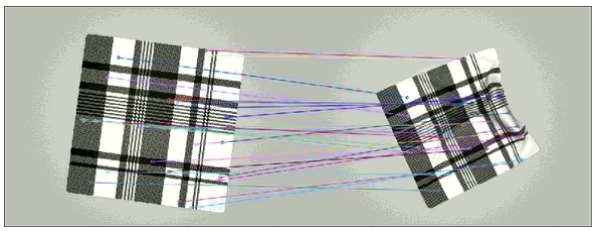


Fig. 10. Correspondence matching of 20 pixel pairs randomly sampled inside the fabric closest in the descriptor space between two configurations of a fabric.

ACKNOWLEDGEMENT

This work was supported by the NSF Smart and Autonomous Systems Program (IIS-1849154), USAF and DARPA under Contract No. FA8750-18-C-0092, LG Electronics, a NSF Graduate Research Fellowship (DGE-1745016), and the CMU RISS. Khush Agrawal would like to thank Rachel Burcin and Prof. John Dolan for organizing RISS and his colleagues at IvLabs and Visvesvaraya National Institute of Technology for supporting him during his research.

REFERENCES

[1] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine, "Skew-fit: State-covering self-supervised reinforcement learning," *CoRR*, vol. abs/1903.03698, 2019. [Online]. Available: http://arxiv.org/abs/1903.03698

[2] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 9209–9220. [Online]. Available: http://papers.nips.cc/paper/8132-visual-reinforcement-learning-with-imagined-goals

[3] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," *CoRR*, vol. abs/1806.08756, 2018. [Online]. Available: http://arxiv.org/abs/1806.08756

[4] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 492–499, 2020.

[5] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Learning to smooth and fold real fabric using dense object descriptors trained on synthetic color images." *CoRR*, vol. abs/2003.12698, 2020. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr2003.htmlabs-2003-12698

[6] Y. Li, Y. Wang, M. Case, S. Chang, and P. K. Allen, "Real-time pose estimation of deformable objects using a volumetric approach," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. IEEE, 2014, pp. 1046–1052. [Online]. Available: https://doi.org/10.1109/IROS.2014.6942687

[7] R. Jangir, G. Alenyà, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning." *CoRR*, vol. abs/1910.14475, 2019. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1910.htmlabs-1910-14475

[8] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," *CoRR*, vol. abs/2003.05436, 2020. [Online]. Available: https://arxiv.org/abs/2003.05436

[9] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning." in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.htmlLillicrapHPHETS15

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

Rock Detection and Accurate Boundary Localization Through Non-Learning Based Superpixel Optimization

Ali Albazroun[1], Raewyn Duvall[2], and William L. Whittaker[2]

***Abstract*— Rock detection is of high importance in lunar surface navigation and study. Many different rock detection methods were formulated over the years. Most recently, region-based methods that segment images into regions that share common characteristics have been used to find rocks and accurately localize their boundaries. However, these recent methods have primarily used learning algorithms to classify the generated regions into rocks or background, and these algorithms require fairly large image sets to train them which are not available for the lunar surface. To compensate for this, we propose the use of a hybrid approach that uses edge-detection to identify rocks in conjugation with superpixel segmentation to accurately localize the boundaries of said rocks by solving an optimization problem. We tested our method on real lunar images from the Chang'e 3 lunar mission and we will be evaluating its performance in the Iris lunar Mission.**

***Index Terms*— Object Detection, Segmentation and Categorization, Space Robotics, Superpixels.**

I. INTRODUCTION

Rock detection is an essential feature for rover navigation and is important in the collection and study of geological features during space missions. While accurate rock detection is of high importance it is difficult to achieve due to the diverse morphological features such as size, texture, and color exhibited by rocks. Multiple vision based techniques have been developed over the last few decades to accurately detect rocks such as edge-based and region-based methods. Recently, region-based methods [1]–[4] have garnered a lot of attention due to their ability to generate accurate rock boundaries. These region-based methods rely on the use of learning algorithms [1]–[4] to classify the regions (most commonly superpixels) into rock or background. However, these classification techniques require large image sets to achieve adequate detection accuracy. This poses a problem in the case of lunar exploration and space exploration in general where such large accurately labeled image sets are not readily available. To address this problem, we propose a hybrid method that uses edge-detection and superpixels to detect rocks and localize boundaries without the need of learning algorithms. To do so we start by creating regions of interest using BLOB (Binary Large OBject) analysis on a binary image produced by the Prewitt edge detection method/range filtering that roughly estimates the shape and the positions of the rocks. Then using a SLIC (Simple Linear Iterative Clustering) based superpixel segmentation [5] of the image we compute the mean features (such as intensity, texture, etc.) in each of the superpixels then use thresholding to create binary images of possible rock outlines which we could use to solve for the best rock outline through an optimization problem.

Fig. 1. An image of rocks on the moon from the Artificial Lunar Landscape Dataset [6]

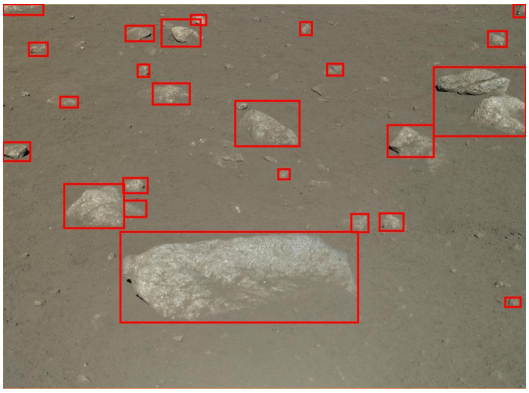


Fig. 2. Bounding Boxes created with the help of the prewitt edge detection method and BLOB detection

II. RELATED WORK

This section is a brief overview of previously used rock detection and outline localization methods.

A. Edge Based Methods

Edge based methods like in [7]–[9] use edge detection algorithms like the Canny or Sobel algorithms to find edges in an image, then a series of edge cleaning and closing

[1]Ali Albazroun is with the Mechanical Engineering Department, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA aia@illinois.net

[2]Raewyn Duvall and William L. Whittaker are with the Field Robotics Center, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {rduvall,red}@andrew.cmu.edu.

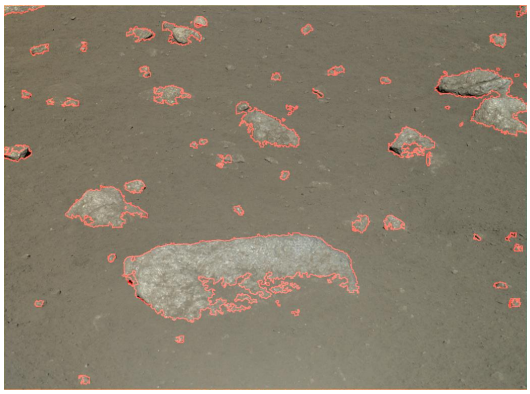


Fig. 3. Rock outlines produced by our optimization method

operations are performed to create a set of closed contours which can be classified as rocks. These methods have the advantage of not relying on learning algorithms which allows them to be used without any prior training while also attempting to localize the boundary of the rock. However, edge detection methods have the possibility of introducing false edges and/or missing edges which could substantially affect the accuracy of the rock boundary localization.

B. Region Based Methods

Region based methods like [1], [3], [4] start by partitioning the image into regions that share similar characteristics then use a classification algorithm to classify each region into rock or background. Most recently these methods have used superpixels which rely on K-means clustering to divide the images into segments and learning algorithms such as AdaBoost [2]. Similar to the edge based methods, region based methods are able to localize the boundaries of the rocks. However, due to their reliance on learning algorithms, they require training which as mentioned before need sufficiently large enough data sets to achieve ample rock detection accuracy.

C. Stereo Vision Based Methods

Stereo based methods rely on the use of LIDARS [10] or multiple cameras [11]–[13] to estimate a ground plane from which significant protrusions are classified as rocks. These methods are not used as much as the previously mentioned methods because they require the use of more hardware such as LIDARS or multiple cameras which come with extra costs, set up and maintenance.

D. Shadow Based Methods

Shadow based methods such as [14] depend on a given sun angle from which the location of rocks could be estimated using their shadows. Shadow based methods are not used as often as the others due to their inability to find the outline of the rocks detected.

Our work attempts to create a hybrid approach between Edge Based Methods like [7]–[9] and Region Based Methods such as [1], [3], [4] that accurately detects rocks and localizes their boundaries while avoiding the use of learning algorithms that require significant training.

III. METHODS

To give an overview of our proposed method, we begin by creating a mask for the ground to simplify the detection process. After that, we use Prewitt edge detection/range filtering in conjunction with Gaussian filtering to create a rough estimate of where the rocks are located and their shape. Using that rough estimate we create bounding boxes for each rock through BLOB analysis, and create a superpixel segmentation of that said box based on the number of pixels in the bounding box. Next, using that superpixel segmentation we can create different possible rock outlines through thresholding. We then rate each outline based on how much it resembles the rough rock estimate we got using edge detection. We then select the outline with the best rating and repeat the process on different versions of the image to improve the outline quality. Finally, we perform some post processing operations to get rid of holes and smooth out the rock boundary.

A. Preprocessing

Before starting the segmentation process we resize the image to double its size, this is done to improve the rock detection and outline localization process. After that we create multiple versions of the image to measure different features of the image such as texture through entropy filters or different color spaces. Performing the optimization process on this different versions of the image will improve the quality of the rock boundaries.

B. Ground Mask

Since we plan on using edge detection in our method, we have to make sure that the edges we detect are not of the boundary of the sky and the ground or of objects that might be in the sky. For this purpose we utilize two methods to create a ground mask to ensure all the objects we detect are on the ground.

1) Thresholding using Otsu's Method: We compute a threshold using Otsu's method [15] that minimizes the intraclass variance between the pixels above and below the threshold. If the sky is present in the image, the threshold is able to divide the image into the ground minus the shadows in white, along with the sky and the shadows in black. We can easily determine the largest object in the binary image (the sky presumably) and remove all other smaller objects (shadows), to get a mask for the sky whose complement is the ground mask. This method has the benefits of being fast and able to detect the shadows in the image at the same time, but it has its downsides. For example, it assumes that a sky is present in the image which is not always the case like in figure 1. Moreover, it assumes that the sky is the largest object in the binary image which is not always the case depending on the camera angle and the how big the shadows are. That's why we are using two methods that are able to complement each other.

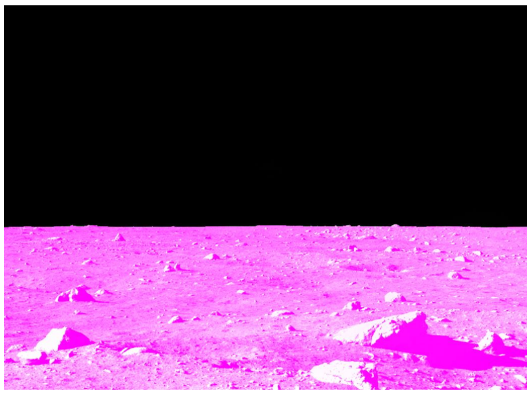


Fig. 4. The ground mask created using Otsu's method of thresholding overlaid on the original image

2) Gaussian Filtering and Edge Detection: An alternative method of determining the ground mask is to use Gaussian filtering on the image to remove the texture from both the sky and the ground then using the prewitt edge detection method to find the horizon line after removing all the smaller edges. If the horizon line is found we take the region below it as the ground mask, otherwise we assume that there is no sky in the image since both methods have failed to find one.

C. Shadow Detection

In this section we detect the shadows in a image in order to improve with outline creation and rock detection process later. Using the ground mask we got from the last section along with Otsu's thresholding method/MATLAB's `imbinarize` we can create a binary mask that outlines the shadows in the image. Note that we performed some morphological opening and closing operations on the image to ensure that the shadows are separated from the sky in case they were connected to each other in the image.

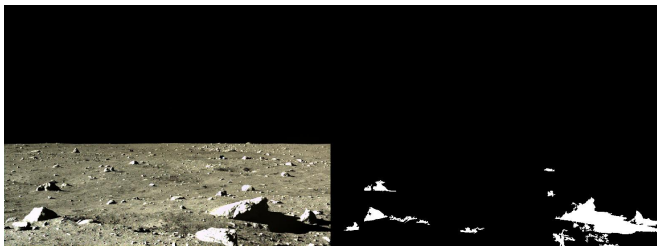


Fig. 5. Shadow mask created using Otsu's method

D. Edge Based BLOB Analysis

To create a rough binary image suitable for rock detection we use edge detection alongside range filtering. First we apply a Gaussian filter to smooth out the grayscale image and remove any unnecessary texture. After that we use the Prewitt edge detection method and/or a thresholded range filter image to detect the rock edges, then apply a Gaussian filter again to smooth out the results and to create a more rounded shape. The result is shown in figure 4. Finally we create a binary image of the result and apply the ground mask to it to make sure the BLOBS detected are on the ground. For the BLOB analysis we used MATLAB's `regionprops` function to detect the BLOBs and create bounding boxes similar to figure 2 for BLOBs of at least a 800 square pixel area and a 25 pixel side length.

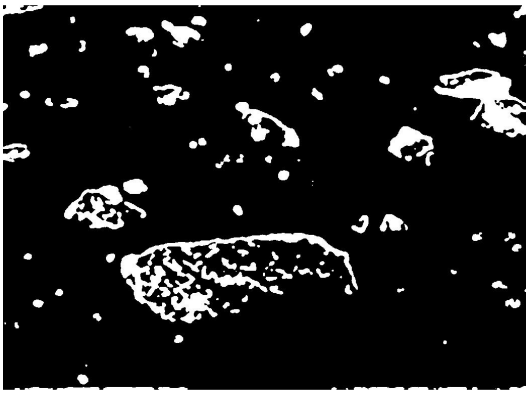


Fig. 6. Rough rock image produced by through edge detection and Gaussian filtering on figure 1

E. Superpixelization

For each bounding box from the previous step we create a superpixel segmentation using the `SLIC0` algorithm [5] to be used for thresholding. We are using a variable number of superpixels depending on the number of pixels in the image to speed up the performance and to ensure that we are not over segmenting the image cropped using the bounding box. The number of superpixels is determined by the following equation:

$$N_{superpixels} = 5\left\lfloor \sqrt{N_{pixels}} \right\rfloor \tag{1}$$

This equation ensures that the number of superpixels will always be less than the number of normal pixels while also having a greater number of superpixels for larger images to ensure it can capture all the details of the rock boundary.

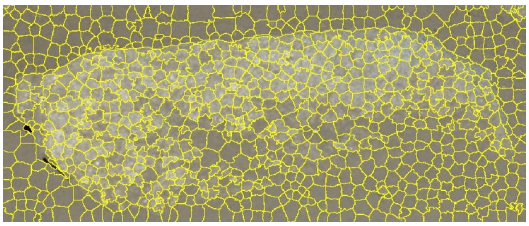


Fig. 7. Superpixels produced by the SLIC algorithm overlaid on the cropped image of a rock from figure 1

F. Threshold Optimization

To determine which superpixels belong to a rock and which do not, we use thresholding on the multiple versions of the image created in the preprocessing step. The determination of the best threshold is framed as an optimization problem where we are maximizing the number of superpixels inside the rock boundary and minimizing the number of

superpixels outside. To do this we created a metric that is able approximate the difference between those quantities. Given a cropped image I using a bounding box with m rows and n columns from a version of the original image and a threshold T. We can create a binary image I_T by assigning a 1 to each superpixel with a mean value greater than the threshold T and a 0 to the other superpixels. To approximate the actual binary image of the rock we can use the rough rock estimate we got using edge detection with the shadows removed from it, which we will denote with I_{BLOB}. We can then perform pixel wise multiplication denoted by $\star$ on each I_T with I_{BLOB} to determine the number of pixels equal to 1 inside the rock boundary N_{in} and its complement $(I_{BLOB})^C$ to determine the number of pixels outside the rock boundary N_{out} by summing over the rows and columns of the corresponding product images.

$$N_{in}(T) = \sum_{i=1}^{m}\sum_{j=1}^{n}(I_T * I_{BLOB})(i,j) \tag{2}$$

$$N_{out}(T) = \sum_{i=1}^{m}\sum_{j=1}^{n}(I_T * (I_{BLOB})^C)(i,j) \tag{3}$$

Lastly, the rating metric $E(T)$ we are maximizing is then:

$$E(T) = N_{in}(T) - r[N_{out}(T)] \tag{4}$$

$$r = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} I_{BLOB}(i,j)}{\sum_{i=1}^{m}\sum_{j=1}^{n}(I_{BLOB}(i,j))^C} \tag{5}$$

where r is a normalization ratio equal to the ratio between the number of 1's in I_{BLOB} over the number of 1's in ${I_{BLOB}}^C$ to ensure that both parts of the expression are weighted equally. This formulation of the problem assumes a threshold value T_{above} where superpixels with mean values greater than T_{above} are assigned a 1, but we could assume that superpixels with mean values less than T_{below} are assigned 1's instead. This definition results in an image that is the complement of I_T from before, ${I_T}^C$. This alternative definition of the threshold could in fact result in a better way to segment the image since it could result in binary images with higher values of $E(T)$. We can address this problem by replacing I_T with ${I_T}^C$ in the optimization function $E(T)$ and simplifying to get the following result:

$$E(T_{above}) = -E(T_{below}) \tag{6}$$

This implies that:

$$max(E(T_{below})) = -min(E(T_{above})) \tag{7}$$

This fact means that we only need to use the first threshold definition, we just need to compare the maximum value of $E(T)$ and the negative of the minimum value of $E(T)$ to find the optimum binary image for the rock outline.

After repeating the optimization process for each version of the image we created in the preprocessing section and finding the optimum threshold for each, we use those thresholds and the corresponding binary images to count how many times each superpixel has been assigned a 1 in a binary image and compare to the average number of times the superpixels have been assigned a 1. If a superpixel has been assigned a 1 more than the average of all the superpixels it is included in the final binary image. This procedure is done to give certainty to the final segmentation because by choosing the superpixels that were assigned a 1 an above average number of times we eliminate most of the superpixels that were assigned a 1 by chance in any of the binary images created by the thresholds.

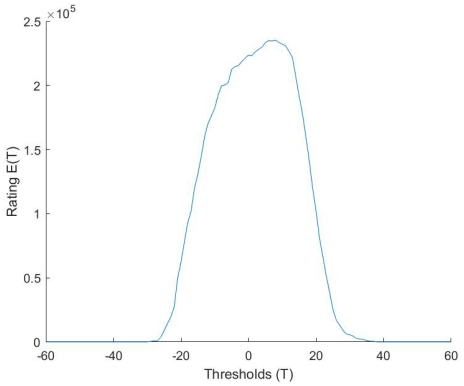


Fig. 8. An example of a rating curve $E(T)$ where the optimum threshold coincides with the maximum value of $E(T)$

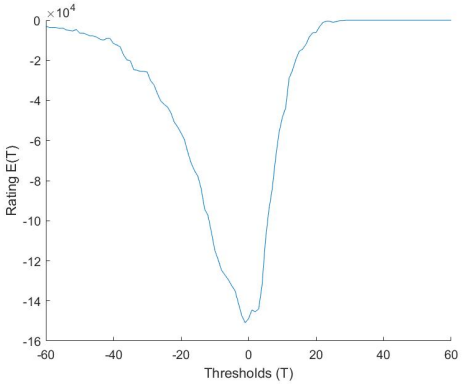


Fig. 9. An example of a rating curve $E(T)$ where the optimum threshold coincides with the negative of the minimum value of $E(T)$

Fig. 10. The binary image created after running the threshold optimization process on all versions of the image

G. Post Processing

To create better looking outline we perform a series of operations to fill any holes and get rid of any small objects present in the binary image.

Fig. 11. The final binary image of the rock after post processing

IV. RESULTS

To test the results of our method we use the real moon images provided in the Kaggle Artificial Lunar Landscape Dataset [6] alongside their ground truth labels for the rocks in the images. The images we used in our testing came from the Chang'e 3 rover's panoramic camera (PCAM) and lander's terrain camera (TCAM). We adopt accuracy criteria similar to the ones presented in [4] to test our method. We use an Intel CORE I5 1.6 GHz processor with 8 GB of RAM.

A. Region Labeling Accuracy

This criterion is calculated by taking the ratio of the number of pixels correctly assigned to a rocks divided by the number of pixels that truly belong to rocks using the ground truth image as a reference. We tested our method on 15 images from the dataset (7 PCAM images and 8 TCAM images).

TABLE I
REGION LABELING ACCURACY RESULTS

Image Source	PCAM	TCAM	Both
Accuracy	81.03%	95.03%	88.50%

We are able to consistently achieve a region labeling accuracy of above 90% for the TCAM images and around 77-88% for the PCAM images. The difference in accuracy can be attributed to the PCAM images being panoramic which lowers the quality of the images.

B. Rock Detection Accuracy

This criterion is concerned with the number of rocks detected correctly in an image. We define a correctly detected rock if it has over 50% area overlap with a rock in the ground truth image. We measure precision (the ratio of correctly detected rocks to all detected rocks) and recall (the ratio of correctly detected rocks to number of rocks in the ground truth image).

Notice that the precision value are very low which we mainly attribute to our method detecting more rocks than the rocks labeled in the ground truth data which artificially increases the number of "false positive" results leading to lower precision values just like in the figure below. Our

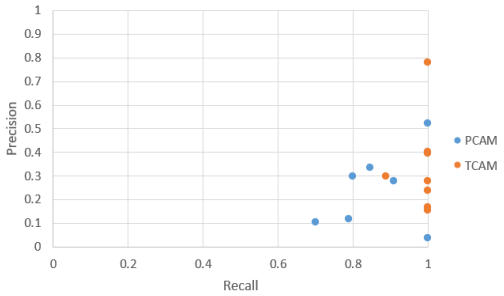


Fig. 12. Rock Detection Precision vs Recall

method achieves on average a recall value of 92.3% which shows that we are able detect almost all the rocks labeled in the ground truth image correctly.

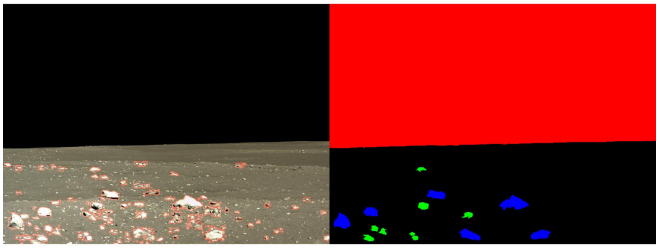


Fig. 13. Rocks detected by our method outlined in red vs rocks in the ground truth image

C. Execution Time

We measure the time it takes to run the code on a single image using MATLAB's `tic` and `toc` to time it.

TABLE II
AVERAGE EXECUTION TIMES

Image Source	PCAM	TCAM	Both
Time (in Seconds)	261.40	144.40	199.00

V. FUTURE WORK

Some adjustments could be made to improve the accuracy and performance of our method. We observed that the accuracy is lowered in grayscale images so in order to improve the accuracy in these images we suggest introducing different versions of the image in the optimization process that are less sensitive to color and more sensitive to other features. Another improvement that could be done is to the execution time. In the testing we observed that a significant amount of time is devoted to creating a version of the image that assigns the mean value of each superpixel to it so it can be used for thresholding. A way that the performance can be improved is by finding a way to threshold the superpixels without creating a new image that assigns the mean value to each superpixel. Also, the number of superpixels used in each bounding box could be further optimized to enhance performance, but not

lose accuracy. Further testing and evaluation of this method will be done in the future during the Iris moon rover mission.

ACKNOWLEDGMENT

This paper was made with the support of Robotics Institute Summer Scholars Program and the KAUST Gifted Student's Program. Ali would like to thank Rachel Burcin, John Dolan and the RISS team for their support during this research project.

REFERENCES

[1] D. R. T. H. Dunlop and D. Wettergreen, "Multi-scale features for detection and segmentation of rocks in mars images," *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[2] H. Q. X. Mu and X. Li, "Automatic segmentation of images with superpixel similarity combined with deep learning," *Circuits, Systems, and Signal Processing*, vol. 39, 09 2019.

[3] S. Niekum, "Reliable rock detection and classification for autonomous science," 2008.

[4] X. Gong and J. Liu, "Rock detection via superpixel graph cuts," in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 2149–2152.

[5] K. S. A. L. P. F. R. Achanta, A. Shaji and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 2274–2282, May 2012.

[6] R. Pessia, "Artificial lunar landscape dataset," 2019, data retrieved from Kaggle on August 2nd 2020, https://www.kaggle.com/romainpessia/artificial-lunar-rocky-landscape-dataset.

[7] T. E. R. Castaño, M. Judd and R. C. Anderson, "Current results from a rover science data analysis system," 2005.

[8] T. E. R. C. A. D. G. A. C. B. B. T. S. R. Castaño, M. Judd and K. Wagstaff, "Current results from a rover science data analysis system," Big Sky, Montana, 2006.

[9] D. G. C. C. B. B. R. C. A. M. B. D. T. R. Castaño, T. Estlin and M. Judd, "Onboard autonomous rover science," Big Sky, Montana, 2007.

[10] R. A. W. E. Johnson, A. Huertas and J. F. Montgomery, "Analysis of on-board hazard detection and avoidance for safe lunar landing," *IEEE Aerospace Conference*, 2008.

[11] T. S. D. Thompson and D. Wettergreen, "Data mining during rover traverse: From images to geologic signatures," *ISAIRAS*, 2005.

[12] R. C. A. J. Fox, R. Castafio, "Onboard autonomous rock shape analysis for mars rovers," Big Sky, Montana, 2002.

[13] R. M. R. C. A. V. Gor, R. Castanfo and E. Mjolsness, "Autonomous rock detection for mars terrain," *Space 2001 (AIAA)*, 08 2001.

[14] M. R. V. Gulick, R. Morris and T. Roush, "Autonomous image analyses during the 1999 marsokhod rover field test," *jgr*, vol. 106, pp. 7745–7764, 04 2001.

[15] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

Evaluating Parameters in Successful Goal from Gaze Predictions

Nadia Almutlak[1], Reuben Aronson[2] and Henny Admoni[3]

***Abstract*— Human- Robot collaboration systems enable people to complete activities that would otherwise be difficult. These systems provide assistance with the knowledge of a user's goals. Recent work has suggested that eye gaze can be a powerful signal in predicting a user's goal early in the progressions of a collaborative task. We previously developed a model to predict a user's goal from their gaze throughout a task. However, this model can be weak when gaze information is not available or inaccurate at different stages of the task. This problem may be exasperated as the robots relying on this system transition to online delivery. The goal of this research was to analyze the relationship between different available parameters, such as distance from the target, and an increase the the accuracy of the gaze prediction model. We evaluated these parameters using previously collected data of users operating a robot in an eating task.**

***Index Terms*— Eye Gaze, Shared Control, Human-Robot Collaboration, HMM, Intent Prediction**

I. INTRODUCTION

Human Robot collaboration systems enable users to complete activities which otherwise might be difficult or impossible. An application of these systems can be found in the domain of assistive robotics. These robots aid in complex tasks like eating, ultimately restoring autonomy to individuals with disabilities. These systems provide useful assistance by engaging both the user's high-level task planning abilities and the robot's precision in a form of shared control to complete these tasks. However, they also rely on the knowledge of the user's goal in the task. [1] A number of approaches have utilized controller input to generate these goal predictions, but are often impeded by the difficulty of using these controllers and teleoperating robots. [1]–[3] This has spurred a growth in the body of work that explores more natural signals in generating goal predictions, specifically eye gaze.

Eye gaze is highly informative signal which produces different types of useful data. [4], [5] As such, more recent work has explored eye gaze as a promising addition to generating goal predictions in a collaboration system. [6], [7] Unlike controller input, eye gaze is continuously provided by the user and does not require explicit input into the system. It also has the ability to indicate the user's goal early in the progression of the task, which the controller input may struggle with. [1] Eye gaze can also be used to indicate difficulties faced by the user in the progression of the task. [8]

While gaze has been explored in predicting a user's goal, it can be a difficult signal to use as it is susceptible to interpretive errors. Natural eye gaze does not provide much context about the scene without undergoing labeling processes. It is also not always a reliable signal as users react to different scenes and tasks with non-identical behaviours. For example, some users may rely on peripheral vision to to gauge the general location of their goal. This approach minimizes the availability of high quality gaze points directed at the user's target, making models that rely on those specific points not as ideal. Users will also exhibit gaze points scattered through the scene as the robot moves and they navigate it to the goal. Here, gaze labels may be misidentified and lead to inaccurate predictions. While these scenarios occur in controlled lab environment, they will only be intensified in different environments as the use of these systems become more common and begin to rely on online delivery.

To alleviate some of those issues, a model is currently being developed to predict the user's goal from their sequential gaze points. [9] However, the accuracy of those predictions need to be further evaluated for the cases where the limitations mentioned above apply. From this ongoing work, there seems to be promising evidence that the progression of the task impacts the accuracy of the goal predictions. It stands to reason that parameters with relation to the progression of the task may be able to act as proxy signal for accurate goal from gaze predictions. In this research, we focus specifically on evaluating how the accuracy of an eye-gaze prediction model changes with respect to these parameters. We ask when is a gaze based prediction most accurate? and what changes in a system are indicators of that accuracy?

To answer these questions, we:

- Defined some initial parameters:
  - Distance to target
  - Robot Mode
- Established a metric for evaluation: The change in probability of accurate goal predictions
- Analyzed the parameters with respect to the goal predictions generated by our model [9] and the HARMONIC dataset [10] according to the metric.

This work is ongoing and the results were inconclusive at the time of this submission.

II. RELATED WORK

This work builds on previous work on shared control with eye gaze [1], [6], [11] It utilizes a prediction model that is

[1]N. Almutlak studies mechanical engineering at Columbia University in New York. na2736@columbia.edu

[2] R. Aronson a Ph.D. student in Carnegie Mellon's Robotics Institute. reubena@andrew.cmu.edu

[3] H. Admoni is an Assistant Professor in Carnegie Mellon's Robotics Institute and leads the Human and Robot Partners (HARP) lab. Henny@Andrew.cmu.edu

currently under development that builds on previous work on gaze based predictions. [7], [9]

A. Shared Control

Shared control is an approach to teleoperation which engages both the user's high-level planning abilities and the robot's precision to complete a task. This collaboration relies on prior knowledge of the user's goal which allows an autonomous system to provide assistance with path planning. [1].

These systems work by continuously updating the probability of a goal's location, and planning with respect to those updates. The predictions that these system rely on have been generated in a number of ways from using Maximum Entropy inverse models to fit a user's actions to models [1] to using partially observable Markov decision process (POMDPs) to generate goal probabilities over a location [12]. These systems have shown improved task success rates, reducing task time and providing better user experiences. They required less physical effort and achieved greater precision with the user's input. As a result, systems using shared control could easily be integrated into robotic arms, wheelchairs and more for better overall experiences [1], [3], [12].

B. Eye Gaze for Goal Predictions

Eye gaze has been extensively documented as a useful source of information. Eye gaze is highly informative signal, with the ability to learn about a scene and convey user's intentions without much effort from the user. [2], [13], [14] When a user is performing a task, they broadcast their intentions to manipulate certain objects based on where they are looking. For example, a person making tea looks at a kettle before picking it up [15], or a person looks at a morsel on a plate before using a robotic arm to spear it with a fork. [2]

These observations in hand-eye correlated behavior has allowed researchers to document patterns associated with gaze and task completion. These patterns and behaviours can signify important information about a task such as what may be important or irrelevent in a scene. [**?**], [4]. These behaviours can also indicate the user's planning as gaze was also established to occur in a dynamic orders useful to the task at hand. [13]

While these behaviors were observed in direct hand manipulation, Aronson et al [2] were able to confirm that these patterns also occurred during robot teleoperation. Considering further work that has documented robot usage and eye gaze [16], the premise that gaze could be a useful prediction source for assistive robots became more popular. [13] A few models have been developed or are currently in development for predicting the user's goal from gaze in a robotic task. One model approached gaze from a categorical standpoint, using the foundation that user's look at their goals more than anything in a scene. [7], [14] While this approach may outperform models that only rely on joystick inputs, it relies on the presence of enough gaze data and does not deliver a prediction from the beginning of the task. This has prompted further development of models that would use sequential gaze data to generate probabilities as a faster and more accurate approach. [9] These models still require evaluation which is what this paper sets out to do.

III. APPROACH

A. Problem Statement

In this work, we try to understand how the model's predictive abilities change as the task parameters change. To do this, we evaluate the incremental probability gain when given a set of accurate goal prediction probabilities along the progression of a task. These gains are calculated using the following metric:

$$\log p_{t+1}(goal) - \log p_t(goal) \tag{1}$$

The results produced by this metric are categorized by the desired parameters that we set out to evaluate.

B. Parameter Selection

In choosing parameters to evaluate as proxies for accurate predictions, it was essential to select parameters that were:

- accessible in both online and offline running systems
- related to the progression of the task

The first parameter we examine is distance from the target. From our ongoing work with the model, there are indications that the probability of an accurate goal prediction is higher with gaze at the beginning of the task than with other prediction models using different signals. [9] An early prediction is key to delivering assistance in the task, making gaze an ideal approach for the predictions. However, since users approach the task differently, time cannot be used as a parameter. Instead, we can select parameters that are analogous to the task progression like the distance from the target.

As the user navigates the robot end effector to the target, they are likely to examine the scene as they plan their path. It would be interesting to see if there are certain points at which these glances happen and in turn increase the probability of an accurate goal prediction. We chose to approach this parameter by using both Euclidean and Manhattan distances to gain insight on how different distances respond to the metrics set worth.

The second parameter is Robot Mode. Early work with gaze and shared control showed that users tend to look at the vicinity of the target when switching modes on their controllers. [2] These controllers allow the user to move in 6 Degrees of Freedom (DOFs), made possible through 3 modes of control, which can be seen in Fig. 1

We anticipated that as the user switches modes and looks at the region, certain modes would reflect higher accurate goal from gaze probabilities. This was assumed to be in the X-Y mode as previous work noted that users relied on seeing the target in that mode the most.

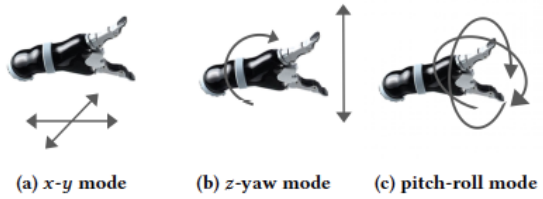


Fig. 1. The three possible Robot Modes and their directions of motion [2]

C. Evaluation

The parameters were evaluated using a goal from gaze prediction model (from [9]) that was based on the HARMONIC dataset. [10]

This dataset features video and gaze data from users working with a Kinova robotic arm on a food-spearing task. The experiment designed in this dataset can be seen Fig. 2, and asks users to spear one of three morsels on a plate using different modes of autonomy including manual teleoperation and shared control. This dataset also contains segmented and filtered gaze data while also providing processed joystick input mode and robot joint information.

From this dataset, we took into consideration the information of 64 teleoperation trials. These trials only included successful attempts at spearing the morsel as to understand how the parameters responses in an ideal scenario.

Fig. 2. Overview of the task in the dataset

A program was constructed to accept input for any type of parameter and output the results defined by our metric.

IV. DISCUSSION

This work presents a preliminary evaluation of how a goal from gaze prediction model's predictive abilities change as the task parameters change. Currently, we have no conclusive results but provide a short discussion on some interesting notes.

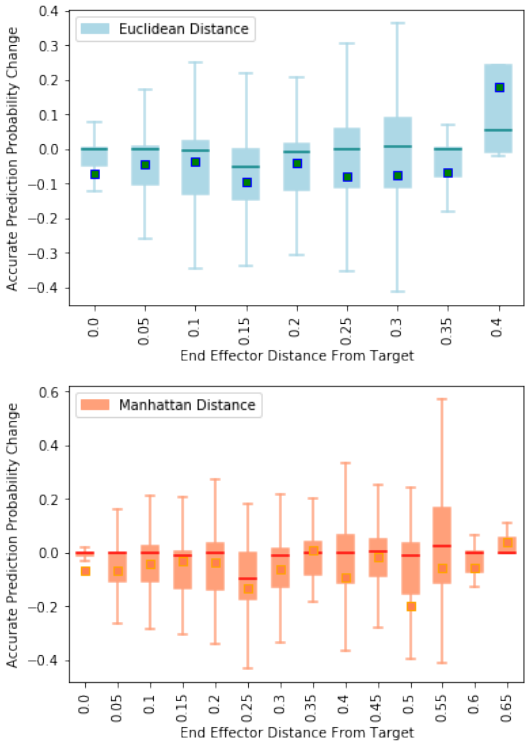


Fig. 3. Distance from the target's interaction with the incremental probability gain for the correct goal

A. Distances

In fig. 3, we can see a general representation of how distance from the target interacts with the probability change for the correct goal. The parameter may be capturing something useful but needs further evaluation. Since distance acts as a proxy for time, its likely that it will reflect the same features that progression of the task will. In a follow up, we will compare the incremental probability gain across the progression of the task and compare it to the distance results. This will either confirm that distance can be considered independently of time or that the two parameters are too similar and not appropriate for our intended use.

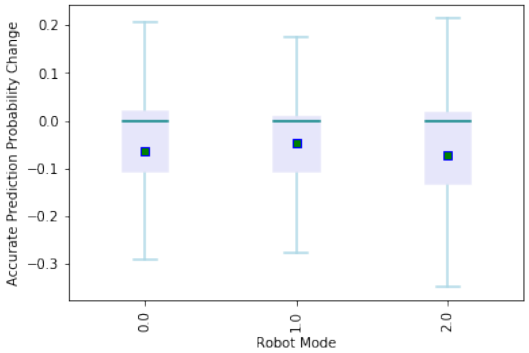


Fig. 4. Robot mode's interaction with the incremental probability for the correct goal

B. Modes

In fig. 4, there is no clear differentiation between the incremental probability changes in any of the mode. This was interesting considering our initial assumption that the X-Y mode would outperform the other modes given its use. This may indicate that on its own, robot mode may not be a reliable proxy. However, it may be interesting to evaluate mode with some other parameter to see if that changes.

ACKNOWLEDGMENT

This work was supported by the Robotics Institute Summer Scholars Program and the KAUST Gifted Student's Program. Special thanks to Reuben Aronson and Dr. Henny Admoni from the Human and Robot Partners Lab at Carnegie Mellon University for their mentorship and support, especially during the summer of Covid-19.

REFERENCES

[1] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, June 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364918776060

[2] R. M. Aronson, T. Santini, T. C. Kübler, E. Kasneci, S. Srinivasa, and H. Admoni, "Eye-Hand Behavior in Human-Robot Shared Manipulation," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction - HRI '18*. Chicago, IL, USA: ACM Press, 2018, pp. 4–13. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3171221.3171287

[3] H. Admoni and S. S. Srinivasa, "Predicting user intent through eye gaze for shared autonomy," in *Proceedings of the 2016 AAAI Fall Symposium: Shared Autonomy in Research and Practice*, November 2016, pp. 298 – 303.

[4] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, "Eye–Hand Coordination in Object Manipulation," *The Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, Sept. 2001. [Online]. Available: http://www.jneurosci.org/lookup/doi/10.1523/JNEUROSCI.21-17-06917.2001

[5] M. Hayhoe and D. Ballard, "Eye movements in natural behavior," *Trends in Cognitive Sciences*, vol. 9, no. 4, pp. 188–194, Apr. 2005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364661305000598

[6] R. M. Aronson and H. Admoni, "Eye Gaze for Assistive Manipulation," in *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. Cambridge United Kingdom: ACM, Mar. 2020, pp. 552–554. [Online]. Available: https://dl.acm.org/doi/10.1145/3371382.3377434

[7] N. Almutlak, R. M. Aronson, and H. Admoni, "Incorporating Eye Gaze into Shared Autonomy for Assistive Robotics," *2019 RISS Working Papers Journal,*, pp. 17–21, Aug. 2019.

[8] R. M. Aronson and H. Admoni, "Gaze for error detection during human-robot shared manipulation," June 2018.

[9] R. M. Aronson, N. Almutlak, and H. Admoni, "Work in progress," 2020.

[10] B. A. Newman, R. M. Aronson, S. S. Srinivasa, K. Kitani, and H. Admoni, "HARMONIC: A Multimodal Dataset of Assistive Human-Robot Collaboration," *arXiv:1807.11154 [cs]*, July 2018, arXiv: 1807.11154. [Online]. Available: http://arxiv.org/abs/1807.11154

[11] H. Admoni and B. Scassellati, "Social Eye Gaze in Human-Robot Interaction: A Review," *Journal of Human-Robot Interaction*, vol. 6, no. 1, p. 25, Mar. 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=3109975

[12] K. Hauser, "Recognition, Prediction, and Planning for Assisted Teleoperation of Freeform Tasks," p. 8.

[13] C.-M. Huang and B. Mutlu, "Anticipatory robot control for efficient human-robot collaboration," in *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, ser. HRI '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 83–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=2906831.2906846

[14] M. Land, N. Mennie, and J. Rusted, "The Roles of Vision and Eye Movements in the Control of Activities of Daily Living," *Perception*, vol. 28, no. 11, pp. 1311–1328, Nov. 1999. [Online]. Available: http://journals.sagepub.com/doi/10.1068/p2935

[15] M. F. Land and M. Hayhoe, "In what ways do eye movements contribute to everyday activities?" *Vision Research*, vol. 41, no. 25-26, pp. 3559–3565, Nov. 2001. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S004269890100102X

[16] K. Sakita, K. Ogawam, S. Murakami, K. Kawamura, and K. Ikeuchi, "Flexible cooperation between human and robot by interpreting human intention from gaze information," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 1. Sendai, Japan: IEEE, 2004, pp. 846–851. [Online]. Available: http://ieeexplore.ieee.org/document/1389458/

[17] E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, and M. Nuttin, "User-adapted plan recognition and user-adapted shared control: A Bayesian approach to semi-autonomous wheelchair driving," *Autonomous Robots*, vol. 24, no. 2, pp. 193–211, Feb. 2008. [Online]. Available: http://link.springer.com/10.1007/s10514-007-9064-5

[18] S. Nikolaidis, Y. X. Zhu, D. Hsu, and S. Srinivasa, "Human-Robot Mutual Adaptation in Shared Autonomy," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI '17*. Vienna, Austria: ACM Press, 2017, pp. 294–302. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2909824.3020252

[19] K. Muelling, A. Venkatraman, J.-S. Valois, J. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy Infused Teleoperation with Application to BCI Manipulation," *arXiv:1503.05451 [cs]*, Mar. 2015, arXiv: 1503.05451. [Online]. Available: http://arxiv.org/abs/1503.05451

[20] C.-M. Huang, S. Andrist, A. Sauppé, and B. Mutlu, "Using gaze patterns to predict task intent in collaboration," *Frontiers in Psychology*, vol. 6, July 2015. [Online]. Available: http://journal.frontiersin.org/Article/10.3389/fpsyg.2015.01049/abstract

[21] B. Ziebart, A. Dey, and J. A. Bagnell, "Probabilistic pointing target prediction via inverse optimal control," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces - IUI '12*. Lisbon, Portugal: ACM Press, 2012, p. 1. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2166966.2166968

[22] A. Fagg, M. Rosenstein, R. Platt, and R. Grupen, "Extracting User Intent in Mixed Initiative Teleoperator Control," in *AIAA 1st Intelligent Systems Technical Conference*. Chicago, Illinois: American Institute of Aeronautics and Astronautics, Sept. 2004. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2004-6309

[23] S. Jain and B. Argall, "Recursive Bayesian Human Intent Recognition in Shared-Control Robotics," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 3905–3912. [Online]. Available: https://ieeexplore.ieee.org/document/8593766/

[24] M. Bernhard, E. Stavrakis, M. Hecher, and M. Wimmer, "Gaze-to-Object Mapping during Visual Search in 3D Virtual Environments," *ACM Transactions on Applied Perception*, vol. 11, no. 3, pp. 1–17, Aug. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2663596.2644812

[25] R. M. Aronson and H. Admoni, "Semantic gaze labeling for human-robot shared manipulation," in *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications - ETRA '19*. Denver, Colorado: ACM Press, 2019, pp. 1–9. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3314111.3319840

[26] M. M. Hayhoe, A. Shrivastava, R. Mruczek, and J. B. Pelz, "Visual memory and motor planning in a natural task," *Journal of Vision*, vol. 3, no. 1, p. 6, Feb. 2003. [Online]. Available: http://jov.arvojournals.org/article.aspx?doi=10.1167/3.1.6

[27] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa, "Assistive teleoperation of robot arms via automatic time-optimal mode switching," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Christchurch, New Zealand: IEEE, Mar. 2016, pp. 35–42. [Online]. Available: http://ieeexplore.ieee.org/document/7451731/

[28] “How do I use Zotero with Overleaf? - LibAnswers.” [Online]. Available: https://libanswers.caltech.edu/faq/204206
[29] Kinova, “Robotic arm series.” [Online]. Available: https://www.kinovarobotics.com/en/products/robotic-arm-series
[30] C.-M. Huang and B. Mutlu, “Anticipatory robot control for efficient human-robot collaboration,” in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Christchurch, New Zealand: IEEE, Mar. 2016, pp. 83–90. [Online]. Available: http://ieeexplore.ieee.org/document/7451737/
[31] A. D. Dragan and S. S. Srinivasa, “A policy-blending formalism for shared control,” *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, June 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913490324
[32] E. C. Grigore, O. Mangin, A. Roncone, and B. Scassellati, “Predicting Supportive Behaviors for Human-Robot Collaboration,” p. 3, 2018.
[33] M. D. Jagli and P. Shetty, “18 Human Emotion Recognition Using Machine Learning,” p. 3, 2019.
[34] K. Mohan, “Automated Facial Expression Detection using Machine Learning Algorithm,” p. 22.
[35] “Recognizing human facial expressions with machine learning,” library Catalog: www.thoughtworks.com. [Online]. Available: https://www.thoughtworks.com/insights/articles/recognizing-human-facial-expressions-machine-learning
[36] M. Saxena, R. K. Pillai, and J. Mostow, “Relating Children’s Automatically Detected Facial Expressions To Their Behavior in RoboTutor,” p. 2.
[37] Y. Wang, Y. Li, Y. Song, and X. Rong, “Facial Expression Recognition Based on Auxiliary Models,” *Algorithms*, vol. 12, no. 11, p. 227, Oct. 2019. [Online]. Available: https://www.mdpi.com/1999-4893/12/11/227
[38] M. Hayhoe and D. Ballard, “Eye movements in natural behavior,” *Trends in Cognitive Sciences*, vol. 9, no. 4, pp. 188–194, Apr. 2005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364661305000598

Kinematic Modeling of The Rolling Star Soft Robot

Samuel C. Alvares*[1], Andrew P. Sabelhaus[2], Carmel Majidi[2]

***Abstract*— Bio-inspired soft robots have flexible limbs that allow them to navigate diverse terrain. A clear model of where the robot and its limbs exist in space is important for it to adapt its shape and locomotion to such varying environments. However, complexities in their mechanics make onboard state estimation of the robot's pose and autonomous closed-loop position control challenging. Such functionality is desirable for trajectory following, disturbance rejection, and more robust behaviors. With the intent of progressing towards the first soft robot capable of onboard state estimation, we develop a novel kinematic model to describe the pose of The Rolling Star, a mobile, soft robot made of seven shape-memory alloy (SMA) arms. The model assumes knowledge (via sensors) of the curvature, as well as the length, of each arm. With these inputs, we model the robot as a single, planar loop of seven rigid bars. Using data from a full dynamics simulation as ground-truth, we demonstrate low error in the model with estimates of less than 2% error through various robot motions.**

***Index Terms*— Kinematics, Soft Sensors and Actuators, Modeling, Control, Learning for Soft Robots, Soft Robot Applications**

I. INTRODUCTION

Robotic systems with high accuracy, precision, and power have long been fabricated using rigid materials [1]. However, these robots are generally very specialized, and their functionality is limited by their rigidity [2]. Bio-inspired soft robots with flexible limbs are capable of multifunctionality far richer than their rigid counterparts. Soft robots can operate in diverse environments, including uneven, narrow, or rocky terrain, or even underwater [3]. In addition, they are much safer for humans to interact with due to their soft limbs which deform upon contact, a concept called *morphological intelligence* [4].

Currently, most soft robots operate without onboard state estimation, meaning that they lack position feedback from sensors [5], [6]. Some progress has been made in the field to implement state estimation and closed-loop position control on a soft robot. Prior work has involved offboard computer vision [7], [8], feedback via acoustics [9], and a number of other efforts [10], [11]. However, none have used onboard sensors for full state estimation of the robot's pose. Such a robot would be progress towards autonomous closed-loop control which permits trajectory following, disturbance rejection, and more robust behaviors.

* *corresponding author.*

[1] S.C. Alvares is with the Robotics Institute Summer School at Carnegie Mellon University, Pittsburgh, PA 15213, USA and also with Rose-Hulman Institute of Technology, Terre Haute, IN 47803, USA salvares@andrew.cmu.edu.

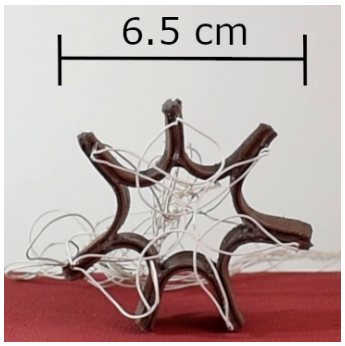
[2] A.P. Sabelhaus and C. Majidi are with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA {asabelha, cmajidi} @andrew.cmu.edu.

In this paper, we discuss how we create a kinematics algorithm to define the pose of a soft robot, an important advancement towards onboard state estimation. We model The Rolling Star, which is formed by a closed loop of seven shape-memory alloy (SMA) arms that actuate with the input of current. Progress in the development of a liquid metal capacitive curvature sensor capable of collecting live curvature data inspired the following kinematic model. Assuming the arms have approximately constant curvature, the sensor data, in combination with the arc-length, allow us to model the robot as a planar, single-loop mechanism composed of seven straight bars connecting the tips of the robot's curved sections. As the limbs of the robot actuate, the lengths of the straight bars change as functions of curvature. This model provides the first closed-form solution to the kinematics of single-loop, closed-chain, N-bar, planar mechanisms. We model the kinematics of the straight sections as rigid bars that bisect the curved sections. Our model demonstrates low error in the locations of the center of mass, the tips of the curved sections, and the outer tips when compared with ground-truth simulation data from a dynamics model of The Rolling Star [12].

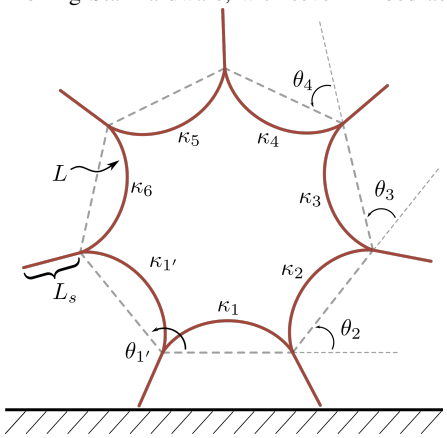
II. BACKGROUND AND RELATED WORK

State estimation, dynamic modeling, and closed-loop position control depend on clearly defined kinematics. Various kinematic models have been created for soft and curved robots [13], but none describe closed-chains of curved limbs, nor do they accommodate curvature as a main input. Kinematic models of straight-bar, planar, closed-chain mechanisms are well-known [14]–[16]. However, we discovered no literature on the kinematics of a single-loop, planar mechanism with greater than six bars [14], [15]. Also, previous models of single loop mechanisms are concerned with stationary robots, and they mainly consider the pose of an end-effector or the location of a single joint.

Huang et al. created a dynamic simulation of The Rolling Star, a mobile soft robot that has previously demonstrated locomotion [2]. They verified their simulation by showing quantitative agreement between their simulation and hardware experiments. The numerical simulation uses a technique called discrete differential geometry (DDG) [12], and the locations of discretized points describing the robot's pose are available as outputs from the simulation. Although the DDG simulation is not applicable to real-time state estimation as it is too computationally expensive, the data output is useful for testing the reduced-order kinematics model we develop.



(a) The Rolling Star hardware, with seven limbed actuators.



(b) Kinematic approximation of the robot, with constant curvatures, κ_i, for each limb and angles, θ_i, between virtual'bars'. The arc-length of the curved sections and the length of the straight sections are represented by L_s and L, respectively.

Fig. 1: The Rolling Star in hardware and its kinematic approximation of curved and straight sections.

III. METHODOLOGY

In light of the previous work on kinematic models of curved robots and on The Rolling Star's dynamic simulation, we aim to develop a novel kinematic model for a soft robot composed of multiple, curved limbs. Since The Rolling Star is a mobile robot, our kinematic model must establish the locations of all external tips to determine environmental contact. The center of mass location must also be calculated because of its importance in future work on control as well as static and dynamic modeling. Additionally, we intend for the kinematics algorithm to be capable of determining the robot's pose in real time on future hardware experiments.

Although The Rolling Star has seven limbs, we develop a more general N-arm model. The model first determines the pose of the curved-section tips, or the points where curved sections join. Then, knowing the curved-section kinematics, it calculates the location of the outer tips of the robot, or the straight-section tips. In the final subsection, we discuss the process for determining the performance of our model.

A. Curved-Section Kinematics

Knowing the curvature and arc-length of a curved section and by making a constant curvature assumption, we can

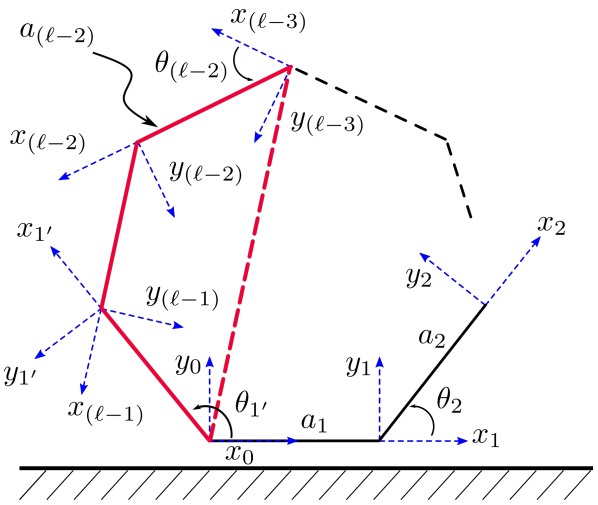


Fig. 2: Curved section kinematics for an N-arm robot. Red lines represent the virtual four-bar mechanism expressed with the final bars, allowing arbitrary N-bar mechanisms to be modeled analytically.

calculate the chord length between the tips of a curved sections, a_i, by

$$a_i = \frac{2\sin\left(\frac{L\kappa_i}{2}\right)}{\kappa_i} \tag{1}$$

where κ_i is the curvature of the limb and L is the arc-length. As motivated by [13], we first convert our model into an approximation of virtual straight bars as shown by the heptagon of dashed, gray lines in Fig. 1b. It is important to note that as the SMA arms actuate, the bar lengths will vary as a functions of curvature.

Considering the curved sections of the robot as a series of straight bars, we establish coordinate frames at each joint in accordance with the Denavit-Hartenberg convention as shown in Fig. 2. We determine the transformation matrices from the base frame to each joint in terms of all joint angles, $\theta_1 ... \theta_{n-1}, \theta_{1'}$, and the bar lengths $a_1 ... a_{n-1}, a_{1'}$, by

$$\mathbf{T}_n^0 = \prod_{i=1}^{n} \mathbf{T}_n^{n-1}, \tag{2}$$

where

$$\mathbf{T}_n^{n-1} = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n) & 0 & a_n\cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n) & 0 & a_n\sin(\theta_n) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

To fully define the pose of the robot, the forward kinematics need to be in terms of a subset of all joint angles, $\theta_1 ... \theta_{n-1}, \theta_{1'}$. Gruebler's formula can be simplified for a single-loop, planar mechanism to

$$F = l - 3 \tag{4}$$

where F is the degrees of freedom of the mechanism and l is the number of links. Thus, to fully define the robot's

pose, the kinematics must be in terms of F joint angles not including θ_1 which is assumed to be 0.

In the simplified case of the four-bar mechanism which has one degree-of-freedom, we can solve for all joint angles in terms a single angle by using law of cosines and law of sines. As shown by the red quadrilateral in Fig. 2, we are able to apply similar mathematics to the four-bar mechanism created by the final three bars of the N-bar mechanism and the bar extending from the base frame to the $(l-3)$ frame. We use law of cosines to solve for the distance between frames $(l-1)$ and $(l-3)$. Law of cosines and law of sines are used to determine unknown internal angles, θ_{l-2} and θ_{l-1}, in terms of F joint angles, $\theta_2 ... \theta_{l-3}, \theta_{1'}$. Setting the transformation matrices in terms of the known joint angles, the pose of each of the curved-section tips is fully defined in terms of the appropriate number of angles.

B. Straight-Section Kinematics

We assume that the straight sections are rigid, have a fixed length, and that they bisect adjacent curved sections at the curved-section tips (Fig. 3). The slope of the bisecting line is

$$m_s = \frac{m_n m_{n+1} - 1 + \sqrt{\left(m_n^2+1\right)\left(m_{n+1}^2+1\right)}}{m_n + m_{n+1}} \tag{5}$$

where m_n and m_{n+1} are the slopes of adjacent curved sections at their junction.

Although two adjacent curved sections come together into one straight section, the curved sections are not assumed to be tangent at their junctions. This constraint would limit the number of other inputs, such as curvatures, that aid in accurately and full defining the pose of the robot's hardware.

Knowing the results of the forward kinematics of the curved sections, the slopes of the straight sections, m_s, and the length of the straight sections, L_s, the location of the tips are

$$\mathbf{x_{t_n}} = \begin{bmatrix} x_n + \sqrt{\frac{L_s^2}{1+m_s^2}} \\ y_n + m_s\sqrt{\frac{L_s^2}{1+m_s^2}} \end{bmatrix} \tag{6}$$

where $[x_{t_n}, y_{t_n}] = \mathbf{x_{t_n}} \in \mathbb{R}^2$.

C. Kinematics Verification

We compare the results of our model to data from the same simulation as was used in [12]. Outputs from the simulation represent the curved and straight sections with discretized points. Because hardware was not available at the time of writing this paper, we create a virtual curvature sensor by performing a least squares estimate and fitting a curve to the curved-sections of the simulation data. By applying law of cosines to the curved-section tips of the simulation data, we are able to determine F joint angles necessary to fully define the robot's pose. The curvature and joint angle calculations occur at each time step, and their results are used as inputs to the kinematic model.

To qualitatively verify our model, we plot an overlay of the simulation and model data as shown in Fig. 4. To quantify

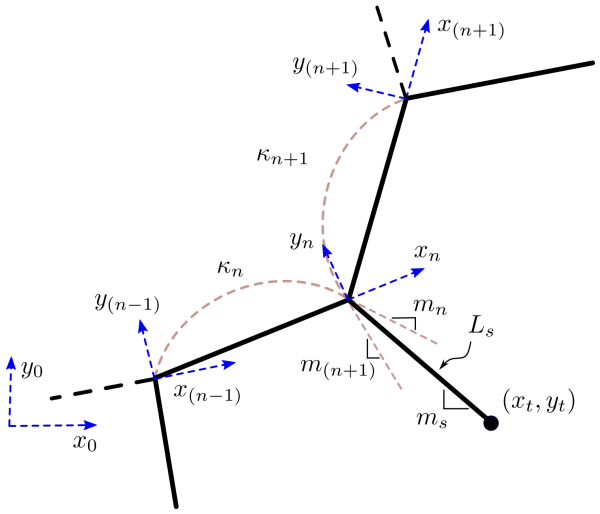


Fig. 3: Straight-section geometry for the nth straight section. Straight sections are assumed to exit the closed chain at an angle, m_s, that bisects the tangent lines to adjacent curved segments.

the error in our model, we calculate

$$\epsilon = ||\mathbf{x}_{sim} - \mathbf{x}_{model}||_2 \tag{7}$$

where $[x_i, y_i] = \mathbf{x}_i \in \mathbb{R}^2$ and $\mathbf{x}_{sim}$ and $\mathbf{x}_{model}$ represent the locations of features from the DDG simulation and from the kinematics model, respectively. By normalizing the error, ϵ, by the height of the robot, h, we get a better sense of the magnitude of model error relative to the size of the robot. The three equations that we use to quantify the results of the model are

$$Error_{com} = \frac{\epsilon_{com}}{h} \tag{8}$$

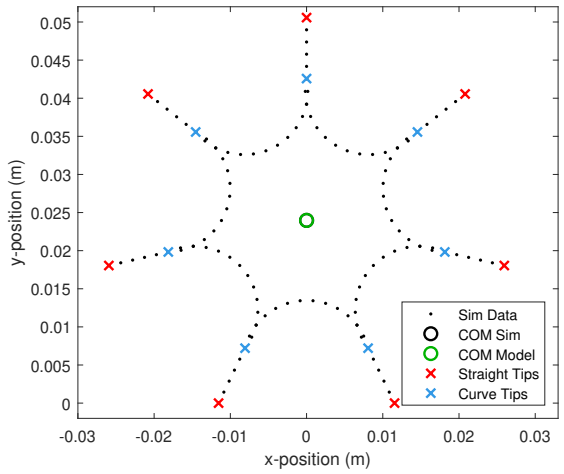
$$Error_{tip} = \frac{\overline{\epsilon_{tip}}}{h} \tag{9}$$

$$Error_{curve} = \frac{\overline{\epsilon_{curve}}}{h} \tag{10}$$

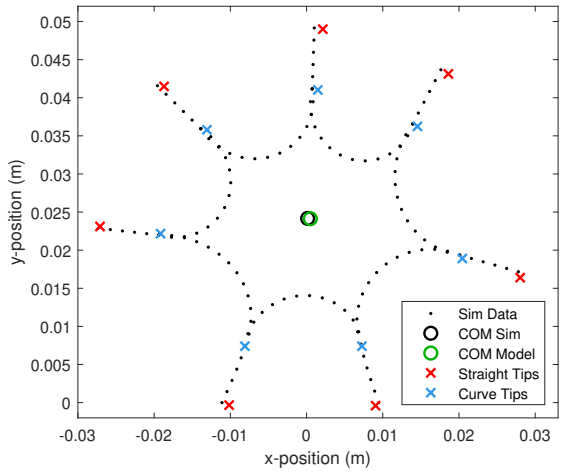
where ϵ_{com} is error in the location of the center of mass, $\overline{\epsilon_{tip}}$ is the average error in the locations of all outer tips, and $\overline{\epsilon_{curve}}$ is the average error of the locations of all curved-section tips, all of which are calculated at one timestep. Note that we calculate the model prediction of the center of mass by averaging the location of the center of mass of each curved and straight section. From the simulation data, the location of the center of mass is calculated by averaging the x- and y-locations of each discretized point.

IV. RESULTS AND DISCUSSION

The outputs from the kinematic model align closely with the DDG simulation with a normalized error in the location of all features less than 2%. Fig. 5 shows a comparison of all error terms throughout the simulation. At the start of the simulation, the robot is not actuated, and gravity has not yet taken effect, so there is little error. Error across all terms increases a small amount shortly thereafter as



(a)



(b)

Fig. 4: Comparison for the kinematic algorithm output to the simulation data (a) at $t = 0$ seconds when no limbs are actuated and (b) at $t = 0.24$ seconds when two of the robot arms are actuating (arms 2 and 1'). Coordinate estimates via the kinematics approximation are low-error in all poses.

gravity deforms the robot. Around 0.24 seconds, two of the arms ($1'$ and 2) are actuating, and all errors spike. As the limbs actuate, the curvatures of the arms likely deviate from constant, and $Error_{curve}$ spikes since the model assumed constant curvature.

There are three main factors that may contribute to the spike in $Error_{tips}$. As the robot's limbs actuate, it has a clockwise rotation. Inertial effects are likely causing the straight sections not contacting the ground to lag behind their predicted locations. In addition, the straight sections likely have slight curvatures induced from inertial effects and from ground frictional contact forces that bend the outer tips away from the model's predictions. Finally, since the locations of the curved-section tips are an input to determine the locations of the straight-section tips, it follows that a spike

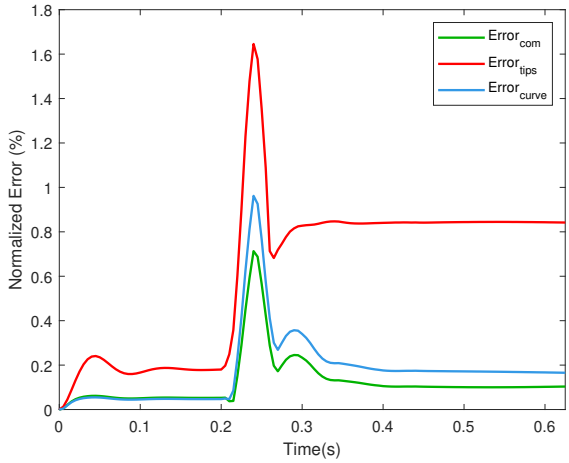


Fig. 5: Comparison of the average error in the location of the straight-section tips, curved-section tips and the error in the location of the center of mass. Error peaks as the robot limbs actuate (0.24 seconds) and various kinematic assumptions become less valid, though errors are still extremely small.

in $Error_{curve}$ contributes to a higher value of $Error_{tips}$.

$Error_{com}$ also spikes when the robot arms actuate. Since the model's center of mass calculation depends on the location of the curved- and straight-section tips, high values of $Error_{tips}$ and $Error_{curve}$ contribute to a high value of $Error_{com}$. The symmetry of the robot causes error in the location of the straight-section tips and curved-section tips to negate, and thus, $Error_{com}$ is consistently lower than the other two error terms.

Towards the end of the simulation, the robot approaches steady state. Due to supporting the weight of the robot and frictional contact forces, the straight sections of the robot contacting the ground likely have induced curvature. Since the kinematic model assumes perfectly rigid straight sections, the error in the average location of the outer tips stays higher than the other terms towards the end of the simulation.

Although we could reduce error by modeling the bending in the straight sections due to contact friction, gravity and inertial effects, error in the location of all features is quite low throughout the entire simulation. Even at the peak of actuation, we see location errors that are well under 2% of the height of the robot, a very small error.

V. CONCLUSIONS AND FUTURE WORK

The development of a soft robot capable of full onboard state estimation of the robot's pose and autonomous closed-loop position control would allow for a wide range of robust behaviors. In this paper, progress towards this functionality by developing a kinematic model for the soft robot, The Rolling Star.

Knowing the curvature of each of the limbs, we simplified the kinematics of the curved sections of the robot by modeling them as straight bars. Then, we determined the location of the outer tips of the robot by finding the bisecting slope

of adjacent curved sections and by knowing the length of the straight sections. The error of our model peaked when the robot actuated two of its limbs. The largest error observed was in the location of the external tips of the robot. In particular, inertial effects and bending in the straight sections were main causes for error in the locations of the outer tips. Any deviation from constant curvature induced error in the location of the curved-section tips. Finally error in the location of the center of mass was the smallest since some of the errors in the robot's features negated due to symmetry. Despite these sources of error, the kinematic model that we presented provides accurate results as measured by a normalized percent error of less than 2% for the center of mass, straight-section tips, and curved-section tips.

To progress closer towards onboard state estimation of this robot, a next step would be to develop a static model. Modeling the robot as a series of straight bars with torsional springs at the joints would allow one to solve for $l-3$ joint angles and achieve pseudo-static state estimation. Another step is to implement this work on hardware which would provide another level of model verification. Liquid metal capacitive sensors could provide curvature readings, and an on-board micro-controller and battery could untether the robot. Also, since The Rolling Star is mobile, another critical step is to model the robot's kinematics as it takes a step, as well as its dynamics.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. This work was also supported by the Carnegie Mellon Robotics Institute. I would like to thank RISS for making the entire program happen online, and I am incredibly grateful for the efforts of Rachel Burcin, John Dolan, and Jennie Piotrzkowski for doing such a fantastic job pivoting to the online environment and leading the cohort. Next, I want to thank my mentor, Andrew Sabelhaus, who committed many hours of his time to my summer experience and to my development as a scholar. I would also like to thank Dr. Carmel Majidi and the rest of the SML for welcoming and integrating me into the lab. Thanks, also, to the members of the Soft Machines Lab for development of the platform studied in this work, as well as the Structures-Computer Interaction Lab at UCLA for the DDG simulations of the robot. From Rose-Hulman, thank you to Dr. Ryder Winck and Dr. Calvin Lui for always believing in me as a roboticist and an engineer. Finally, I would like to thank my family who inspires me to be my best each day.

REFERENCES

[1] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, May 2015. [Online]. Available: http://www.nature.com/articles/nature14543

[2] C. Majidi, "Soft Robotics: A Perspective—Current Trends and Prospects for the Future," *Soft Robotics*, vol. 1, no. 1, pp. 5–11, Mar. 2014. [Online]. Available: https://www.liebertpub.com/doi/10.1089/soro.2013.0001

[3] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, Jan. 2008, publisher: IOS Press. [Online]. Available: https://content.iospress.com/articles/applied-bionics-and-biomechanics/abb355954

[4] C. Paul, "Morphological computation: A basis for the analysis of morphology and control requirements," *Robotics and Autonomous Systems*, vol. 54, no. 8, pp. 619–630, Aug. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889006000613

[5] H.-T. Lin, G. G. Leisk, and B. Trimmer, "GoQBot: a caterpillar-inspired soft-bodied rolling robot," *Bioinspiration & Biomimetics*, vol. 6, no. 2, p. 026007, Apr. 2011, publisher: IOP Publishing. [Online]. Available: https://doi.org/10.1088%2F1748-3182%2F6%2F2%2F026007

[6] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A Resilient, Untethered Soft Robot," *Soft Robotics*, vol. 1, no. 3, pp. 213–223, Aug. 2014, publisher: Mary Ann Liebert, Inc., publishers. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/soro.2014.0008

[7] Z. J. Patterson, A. P. Sabelhaus, K. Chin, and C. Majidi, "An Untethered Brittle Star Robot for Closed-Loop Underwater Locomotion," *arXiv:2003.13529 [cs]*, Mar. 2020, arXiv: 2003.13529. [Online]. Available: http://arxiv.org/abs/2003.13529

[8] W. L. Scott and D. A. Paley, "Geometric Gait Design for a Starfish-Inspired Robot Using a Planar Discrete Elastic Rod Model," *Advanced Intelligent Systems*, vol. 2, no. 6, p. 1900186, 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.201900186. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.201900186

[9] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Science Robotics*, vol. 3, no. 16, p. eaar3449, Mar. 2018. [Online]. Available: https://robotics.sciencemag.org/lookup/doi/10.1126/scirobotics.aar3449

[10] A. D. Marchese, R. Tedrake, and D. Rus, "Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA: IEEE, May 2015, pp. 2528–2535. [Online]. Available: http://ieeexplore.ieee.org/document/7139538/

[11] S. I. Rich, R. J. Wood, and C. Majidi, "Untethered soft robotics," *Nature Electronics*, vol. 1, no. 2, pp. 102–112, Feb. 2018, number: 2 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41928-018-0024-1

[12] W. Huang, X. Huang, C. Majidi, and M. K. Jawed, "Dynamic simulation of articulated soft robots," *Nature Communications*, vol. 11, no. 1, p. 2233, May 2020, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-020-15651-9

[13] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, Nov. 2010. [Online]. Available: https://doi.org/10.1177/0278364910368147

[14] D. Zhang, *Parallel Robotic Machine Tools*. Springer Science & Business Media, Dec. 2009, google-Books-ID: JxeIVEW7oysC.

[15] A. M. Nejad, M. Madani, M. Moallem, and R. V. Patel, "Design and Optimization of a Closed-Chain Mechanism for Constrained Position and Force Control," *IFAC Proceedings Volumes*, vol. 37, no. 14, pp. 693–698, Sept. 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667017311849

[16] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, Aug. 2010.

Continuous Design Variable Optimization in Modular Robot Design through Deep Reinforcement Learning

Max Asselmeier[1], Julian Whitman[2] and Howie Choset[2]

***Abstract*— Modular robots allow for a robust method of catering a robotic system to the task, or tasks, that it is to complete. However, many of the methods that develop ways to generate modular robot designs do so with a finite, discrete pool of modules to pick from. The methods that are able to handle continuous design parameters for these modular arms do not currently leverage the efficiency afforded by deep reinforcement learning algorithms. Continuous design variables would offer another level of versatility and customization with regards to the creation of modular robotic systems. Additionally, reinforcement learning is a computationally efficient way of evaluating modules that can be added to an existing modular arrangement, which is normally an exponentially complex problem. In this work, we offer forth a framework that allows for the combination of the discrete decision of selecting a module group to add to an arrangement through a Deep-Q Network with a continuous decision that optimizes the design variables for the given module group through the Soft Actor-Critic algorithm. We then provide results for the training of the Deep-Q Network on a set of finite modules along with the training of the Soft Actor-Critic algorithm on a relaxed constraint problem.**

***Index Terms*— Kinematics, Novel Deep Learning Methods, Reinforcement Learning, Task Planning**

I. INTRODUCTION

The modular design of robotic arms allows for the specialization of a robot to the task that it is to complete. However, this specialization hinges upon the ability to identify an optimal design, or the proper sequence of modules used to create a robotic arm. This process allows for an expansive amount of creative choices and a high degree of optimization with regards to the specific application of a robotic arm: the more modules that are available to use, the more types of arms that can be generated and the more types of tasks that can be completed. Human experts are capable of generating these optimal robotic arm designs, but this greatly restricts the types of users that can create these arms. Developing a tool capable of learning how to design a modular arm would allow a layperson to engage with this design process and potentially even create a design that an expert user could not conceive of. Additionally, this tool would be useful in situations where the task at hand changes frequently over time. Each unique task would require a re-design from an expert user whereas this tool would be able to produce a useful design in a much quicker fashion.

[1]Max Asselmeier is with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA ma53@illinois.edu

[2]Julian Whitman and Howie Choset are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA jwhitman, choset@cmu.edu

The recent advancements in deep learning have allowed for the utilization of reinforcement learning (RL) as a tool for facilitating search problems such as modular robot design [1], [2]. Computational complexity and time are severe limitations to how large search trees can be, and deep learning provides heuristics for solving these search problems. For modular robot design, the variations of the arrangements of modular arms grow exponentially with the number of different modules that can be added, but deep reinforcement learning provides ways to evaluate module arrangements and focus on those that may be of interest. With this being said, reinforcement learning tools like neural networks also have limitations on how they generate these evaluations. Prior work [1] on this project has employed the use of a Deep-Q Network (DQN), a popular tool in reinforcement learning which requires a finite output space. This type of network provides state-action values known as Q-values that function as scores for each possible action that can be taken from a given state. This works well for modular robot design, but only if we sample from a discrete pool of modules. If we want to adjust design parameters such as the length or mass of a link, then we will end up with an infinitely large action space with each action representing a link of an infinitesimally smaller or larger length or mass, and our problem quickly becomes intractable. This option to tweak design parameters for a robotic arm provides more freedom and flexibility for automated modular robot design, but a different approach that is specifically designed for continuous action spaces must be used.

In this paper, we build on prior methods of modular robot design [1] and develop a framework that would allow for the utilization of a continuous action space. We also initiate work on training a deep RL algorithm that can output the optimal continuous design variables for a certain task. Our framework involves a hierarchical structure of neural networks separating the task of adding a new module to an already existent arrangement of modules into two sub-tasks. A primary network would first choose the discrete type of module to be added to a modular arm such as a link or bracket. Then, a secondary network would set the module type's corresponding design variables. Setting these design variables could be a discrete decision such as choosing if a bracket will be pointed either towards or away from the robotic arm, or it could be a continuous decision such as setting the length of a link. The ability to set or alter these continuous design variables, and more so to do this in the same networks that set the discrete design variables, would allow for a much greater breadth of options when it comes

to which modules are being added to the robotic arm and what characteristics these modules possess.

In this paper, Section II will detail literature and concepts that are related to the work discussed in this paper. Section III will explain the potential methodology of our hierarchical neural networks along with supplementary information for our tasks and solutions. Section IV will present our results, Section V will involve closing remarks on this research, and finally, Section VI will provide limitations of this work along with directions for future research.

II. BACKGROUND

A. Related Works

Previous implementations of modular design synthesis exist, but these methods typically utilize a discrete set of modules. Tools ranging from interactive design systems to best-first graph searches make use of libraries of standard modular components from which complete arrangements for modular robots are created [2], [3], [4]. Evolutionary algorithms are also used to compose robot designs from a finite set of modules [5]. However, these methods all select from a finite number of modules. Genetic algorithms have been able to optimize a mixture of both discrete and continuous design variables [6], but this work does not leverage the deep learning algorithms frequently used now.

Deep learning within robot design has been employed in other forms. For instance, deep learning tools have been utilized to jointly learn not only the structure or design of a robot, but also the robot's motion control policy [7], [8]. Also, prior work on this project has involved utilizing a Deep-Q Network to generate efficient designs for modular serial manipulators [1]. The synthesis of robot designs from a discrete pool of modules allows for the development of robotic systems that are customized towards their given tasks. However, confining modular design synthesis to a set of modules does enforce restrictions on both the types of robots that can be created and the types of tasks that can be accomplished. Continuous design variables would allow for much more freedom when it comes to creating modular robots, and multiple deep learning algorithms that are applicable to continuous action spaces have already been developed.

B. Deep learning in continuous output spaces

Several deep learning algorithms are able to be utilized within continuous output spaces. One way that these algorithms can produce continuous outputs is through methods such as using soft bounding functions to limit the regular outputs of a network's layer to the desired bounds of a given action. The Deep Deterministic Policy Gradients (DDPG) algorithm [9], the deep version of the previously developed Deterministic Policy Gradients [10]. DDPG is a model-free, off-policy algorithm that learns a deterministic policy by using the aforementioned bounding functions. The Twin Delayed Deep Deterministic (TD3) policy gradients algorithm is a modification to DDPG that also does this.

Algorithms can also generate these continuous outputs by producing the mean and standard deviation for a distribution and sampling from this distribution to obtain values. The Soft Actor-Critic (SAC) algorithm [11] is a model-free, off-policy algorithm that performs this idea of sampling from a distribution.

Prior work has used these deep learning algorithms within continuous action spaces to have robot platforms learn simple and compound abstract tasks [12] as well as learn complex manipulation tasks [13]. Deep RL has also been used to learn motion planners with continuous outputs for robotic systems as well. However, no prior work has incorporated continuous action spaces into modular robot design. Our work on this project has initiated the exploration of the optimization of continuous actions within modular design synthesis.

C. Deep learning for Modular Robot Design

Our modular design problem is treated as a finite Markov Decision Process where modules that are to be serially added to an arrangement are evaluated based on the current arrangement as well as the goal position of the episode. The module currently being added to the arrangement is connected to the module that was previously added.

Therefore, the state s_t of this problem is comprised of the active arrangement of the arm, or what modules are currently within the arm along with the goal position that is to be reached. An action a_t is referred to as the process of appending a new module onto the arrangement, and a reward r_t is obtained from the environment at each step based on both the state and the action that is taken. Further information on discrete Q-learning for modular robot design can be found in the previous work on this project [1].

The Soft Actor-Critic algorithm is employed in this project to begin work on the implementation of deep RL on continuous design variables in modular robot design. While most other algorithms only attempt to maximize the expected rewards achieved throughout training, SAC seeks to maximize the expected reward of the actor while also maximizing the entropy of the actor. This means that SAC attempts to have the actor succeed at the given task as frequently as possible while varying its actions as much as possible as well. This idea is evident when viewing the objective function used for the previously discussed algorithms such as DDPG or TD3

$$J(\pi) = \Sigma_t \mathbb{E}_{(s_t,a_t)\sim p_\pi}[r(s_t,a_t)], \tag{1}$$

where s_t, a_t, and $r(s_t,a_t)$ are our state, action, and reward respectively at a certain step, and p_π is our policy. This objective function can then be compared to the augmented objective function that is used for SAC

$$J(\pi) = \sum_{t=0}^{T-1} \mathbb{E}_{(s_t,a_t)\sim p_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))]. \tag{2}$$

We can see that an entropy term denoted by $\mathcal{H}$ is added to the reward, and the importance of the entropy term is decided by the temperature parameter α. If α is set to zero, then the traditional objective function based solely on expected rewards is recovered.

With our objective function in mind, the soft Bellman Equation that is used to estimate the state-action value function Q^π can be formulated as:

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p_s}[V^\pi(s_{t+1})] \quad (3)$$

where the soft value $V^\pi(s_t)$ is equal to

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi}[Q^\pi(s_t, a_t) - \alpha \log \pi(a_t|s_t)]. \quad (4)$$

Here it can be seen that the entropy term for SAC is defined as the negative log of the current policy.

SAC adapts the double-Q learning trick used in TD3 where two critic networks are trained. Once an action is taken, both critics will evaluate the action and return their respective Q-values. When the networks are being trained, the lesser of the two Q-values between the two critics is taken to reduce overestimation bias. SAC also uses target networks [14] and experience replay [15] to facilitate and stabilize the training of the networks. Furthermore, we use Hindsight Experience Replay (HER) [16] which is a data augmentation technique applied to the replay buffer during training to help with the sparse reward function that is used as part of this work.

III. METHODS

This project adapts much of its framework from prior work on this project [1]. We previously utilized a DQN to approximate the state-action values for a set of modules that can be added to a robotic arm. However, since one of the core ideas of this project is the implementation of continuous action spaces which are incompatible with DQNs, the framework that we developed plans on using our DQN in a slightly different way. For this framework, our DQN would instead select the *type* of module to be added. For instance, instead of selecting a link with predetermined design variables, the DQN selects the overall module group of "link", and the variables would be set later on in the architecture of the problem. This combination of discrete and continuous decisions was not achieved during this project, and plans to implement it are detailed in section VI. The desired framework for this hierarchy is detailed below.

The DQN and SAC networks will still be used to design an optimal modular arm design given a goal position in space. The inputs to the networks would aim to represent the current state of the arm along with the goal position that the arm is to reach. The outputs of these networks would aim to select the best possible module that could be appended to the arm in order to reach the goal position. A reward is given to the networks if the arm is able to get within a certain distance to the goal position, and the networks selects modules to maximize both the amount of rewards obtained along with the randomness of the modules chosen.

The task space for this problem is limited to one position $p \in \mathbb{R}^3$ where p = [p_x,p_y,p_z]. This means that during each episode of training, inverse kinematics (IK) is performed on the end-effector of the arrangement with the respective point p of that episode as the goal position. Inverse kinematics is solved through the module PyBullet [17] with the Damped Least Squares method [18]. A tolerance ϵ_p is set so that a reachability function for an arrangement $A \in \mathbb{R}^{N_{max} \times N_m}$ where N_{max} is the maximum number of modules allowed in an arrangement and N_m is the number of modules to choose from and target position $T \in \mathbb{R}^3$ can be defined as follows:

$$reach(A, T) = \begin{cases} 1 & || \, p - p_{EE} \, || < \epsilon_p \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Where p_{EE} is the location of the end-effector of the arrangement in space after forward kinematics has been performed on the arrangement using the angles obtained through inverse kinematics. These modular arrangements are also evaluated on other non-terminal conditions such as the mass and complexity of the arrangement. The mass of the arrangement $M(A)$ is simply calculated by summing up the masses of the individual modules in the arrangement, and the complexity of the arrangement is represented by the number of actuated joints in the arrangement $N_J(A)$. An objective function for these conditions can be defined as

$$F(A, T) = -w_J N_J(A) - w_M M(A) + reach(A, T). \quad (6)$$

Weights w_J and w_M are determined by the user based on how important the mass and complexity of the arrangements are. Therefore, it follows that an optimal arrangement is capable of maximizing this function for its singular goal position:

$$A^* = \arg\max_A F(A, T) \quad (7)$$

Now, we outline the proposed way to train these networks to discover and select these optimal arrangements.

A. DQN for module group selection

The actions from our DQN are chosen from a set of four options: actuators, brackets, links, and end-effectors. As of now, the links are the only module group that support continuous design variables. For the other three module types, the DQN simply would select from a discrete pool of modules with predetermined design variables.

The arrangement of each modular arm would be represented through a list of one-hot vectors such that each index of a vector indicates either the discrete module chosen for the arrangement or the module group selected in the case of a link.

The design variables for the arrangement would be represented through a list of vectors where each index of the vectors represents a singular design variable for the module that is at the position in the arrangement of the corresponding index of the vector within the list. If a module within the arrangement were to be chosen from a discrete set and therefore not possess any continuous design variables, then its vector of design variables would simply consist of all zeroes. Also, if an arrangement would be to end with less than the maximum number of modules, then the vectors occupying the empty indices of the list would also contain all zeroes.

The design variables would be passed through a pre-processing layer for the module type that is to be added so

that even if the numbers of design variables for two types of modules are different, the outputs of the pre-processing layer are still the same size. For instance, a pre-processing layer for links would accept as its input the design variables for a link, a pre-processing layer for brackets would accept as its input the design variables for a bracket, and both of these layers would have the same size outputs. All of these outputs can be appended for a given arrangement so that the processed design variable list is the same size for all arrangements. The structure of the networks is further detailed in Figure 1.

At each step within an episode, the DQN would either select a discrete module or a module group. If a discrete module is chosen, then this module would just be appended to the arrangement and there would be no more activity from the networks for the duration of this step. However, if a module group were to be chosen, then the state of the arrangement along with the module group to be added would be passed to the SAC networks.

The reward function would be identical for all arrangements and it would consist of non-terminal and terminal components. The non-terminal penalties would come from the mass and complexity of the module m that is to be added to the arrangement:

$$r(m) = -w_J N_J(m) - w_M M(m) \tag{8}$$

If the module that is chosen by the DQN were to be an end-effector, then the action as well as the next state of the arrangement would be terminal, and a terminal reward would be returned. The terminal reward function evaluates two features of the arrangement. If the maximum number of modules for an arrangement is reached without an end-effector being added to the arm, then a terminal reward of -1 would be returned. Otherwise, if an end-effector were to be added at any point along the arrangement, then the previously detailed reachability function for the arrangement would be evaluated

$$r_{terminal} = \begin{cases} -1 & \text{length(A') == } N_{max} \\ & \text{and } m \text{ is not an EE} \\ reach(A', T) & m \text{ is an EE} \end{cases} \tag{9}$$

where A' is the arrangement after the action has been taken and EE is an end-effector.

B. SAC for design variable selection

SAC consists of an actor-critic framework where a single actor network learns a stochastic Gaussian policy by outputting mean and standard deviation values that can be used to create a Gaussian distribution. Actions are then sampled from this distribution. Two critic networks evaluate and return Q-values for the actions taken by the actor network. The actor network accepts the state as its input and outputs actions, and the critic networks accept the state as well as the current actions and outputs a Q-value that evaluates the action taken.

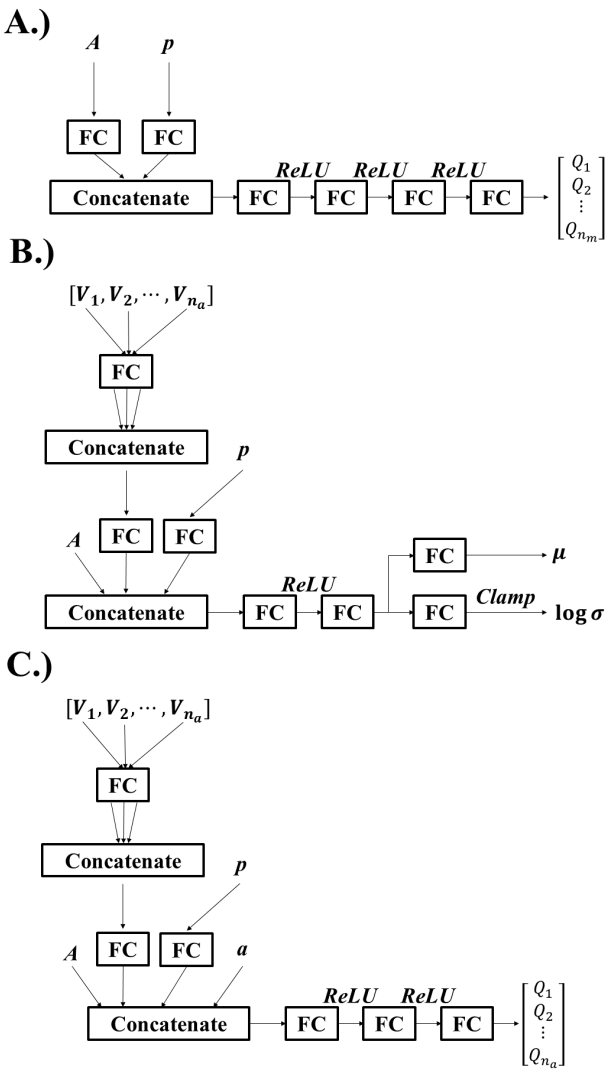


Fig. 1. The structures that we used for our DQN and SAC networks. A.) shows our DQN, B.) shows the actor network for SAC, and C.) shows the critic network for SAC. These networks use fully connected (FC) layers, concatenations, rectified linear units (ReLU), and clamps between minimum and maximum values. The DQN accepts an arrangement A and a target position p as its input and outputs a Q-value for each discrete module or module group. The Q-values span the number of modules or module groups N_m. The SAC actor takes A and p as inputs as well, but also takes in our processed design variables V of length n_a, the maximum number of design variables in an arrangement. The actor outputs a mean μ and a logarithm of a standard deviation $\log \sigma$ which are both used to calculate the action. The critic for SAC takes in the same inputs as the actor while also taking in the action a chosen by the actor. The critic outputs Q-values for each design variable set by the action.

The state for the SAC algorithm would consist of the same elements as the DQN: the current arrangement of the arm along with the goal position that the arm is to reach, but then we would also incorporate the design variables that are set for each module.

In our work, the SAC networks were not configured to train on the IK problems, and they were instead trained on a variable range problem where the sum of the lengths as well as the sum of the twists of all the links in the arrangement

had to fall within a randomly generated range. The twist of a link is the angular difference between the two joint axes at both ends of the link.

Since our SAC networks have only been adapted to links within our arrangements, the reward function only consists of one non-terminal component which penalizes for the mass of a link. The mass is penalized to encourage lighter and cheaper arrangements.

$$r(m) = -w_M M(m) \tag{10}$$

C. Training the neural network architecture

For this project, the DQN was trained on a finite module set to determine that the network functioned properly. At the beginning of each episode, the target position is generated from a random uniform distribution. The X and Y indices of the target position are generated from the range [-0.5, 0.5] whereas the Z index is generated from the range [0.0, 0.5].

As an episode progresses, the arrangement grows by appending a module after each step. At each step, the DQN outputs Q-values for each possible discrete module. The proposed continuous framework would have the DQN instead output a Q-value for each module group. The masking of certain actions is performed to ensure that each type of module can only connect to a certain subset of modules. For example, two actuators cannot be connected and two non-actuators can be connected. Q-values are learned for all actions, but only the Q-values for valid actions are evaluated when selecting an action. The Boltzmann exploration strategy is employed for the DQN in order to handle exploration and exploitation [19]. Often times when building an arrangement for a modular robot, multiple modules represent valid additions that can lead to high-reward states. With this in mind, a method such as ϵ-greedy fails to account for the exploration of multiple valuable actions since ϵ-greedy will either pick an action at random or choose the single most valuable action. Boltzmann exploration makes the process of choosing an action stochastic by creating a probability distribution across all valid actions. This allows higher value actions to still be selected more often while also ensuring that no single action is repeatedly exploited. A temperature parameter can also be adjusted to make the probability distribution more or less skewed towards higher value actions.

For the SAC networks, the actions are bounded by predefined action limits. The length of a link is restricted to the range of [0.0, 0.75] and the twist of a link is restricted to the range of [0.0, 2π]. Exploration is encouraged through the entropy term that is added to the objective function of the SAC networks, and exploration is automatically added through the random sampling of the Gaussian policy that is generated by the actor network.

While training the SAC networks, actions are sampled randomly from a uniform distribution for the first predefined number of steps. This is done to ensure a proper amount of initial exploration and to also allow for more uniformity with respect to the initial weights of the network across separate trials.

A replay buffer is utilized to allow for off-policy learning. At each step, the state, action, reward, and next state are all added to the replay buffer along with a variable signaling if the action taken was a terminal one. Since the reward function for our IK-based tasks is quite sparse, hindsight experience replay (HER) is implemented [16] to ensure that high-reward states are always existent within the buffer that training batches are sampled from. If an episode terminates and the end-effector of the arrangement is not within the distance threshold ϵ_p of the goal position to earn the terminal reward of one, then the same exact tuples of the state, action, reward, next state, and terminal variable are added to the replay buffer again, but now with their goal position as the point in space that the end-effector ended up reaching. However, if the final position of the end-effector lies outside of our original goal ranges, then the tuples are not added to the replay buffer.

Validation checks are also made at certain intervals of training to determine how the policy is being updated throughout training. These validation checks are made by simply having the policy construct an arrangement with solely exploitative actions.

IV. RESULTS

The results of this project contain results obtained from training the DQN on a finite set of modules as well as results from training the SAC networks on a relaxed problem that requires the design variables of the links within an arrangement to fall within certain goal thresholds. Both of these training series occurred separately, and the DQN and SAC networks were not trained together.

Figure 2 shows the total rewards earned from the DQN arrangements during training. These results were tabulated over three trials. It can be seen that for roughly the first 500 episodes of training, the total rewards are negative. This is due to the fact that at the beginning of our training, our Boltzmann exploration tends to select actions uniformly, so modules that are less helpful for reaching a goal position will be selected more often. Also, the network has only just started to train, and it has not seen enough high-reward states to learn what arrangements are able to reach the goal positions.

Figure 3 demonstrates the total rewards earned for the simplified SAC training. These results were also obtained over three trials, and similar trends to the rewards for the DQN training can be observed. The total rewards are negative for roughly the first 500 episodes for the same reasons provided for the DQN training, and soon after this the networks are able to select states that earn positive rewards which leads to a positive, linear increase in total rewards for the remainder of the training.

Figures 4 - 8 demonstrate the evolution of arrangements throughout the validation checks. Earlier arrangements are frequently much too short to reach goals, and as more training episodes are done, these arrangements tend to have the proper number of modules to allow the end-effector to reach the goal. Later on in training, the DQN also learns to

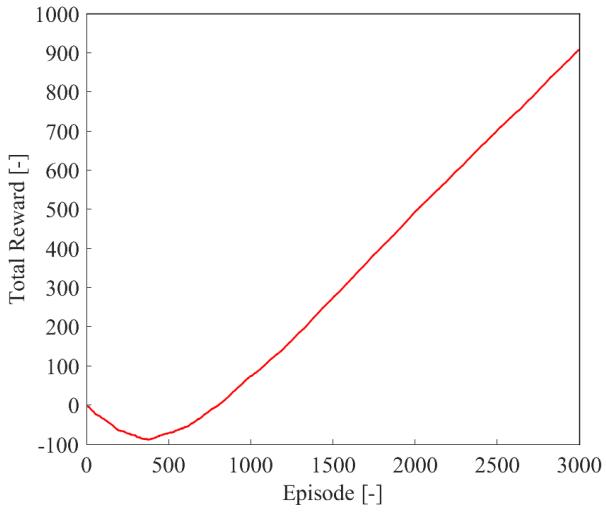


Fig. 2. A plot of the total rewards earned from the DQN over all of the training episodes. These results are averaged over three training trials, and a reward of one was given if the arrangement was able to get within two centimeters of the goal position while minor penalties were given for the mass and complexity of the arrangement at all steps.

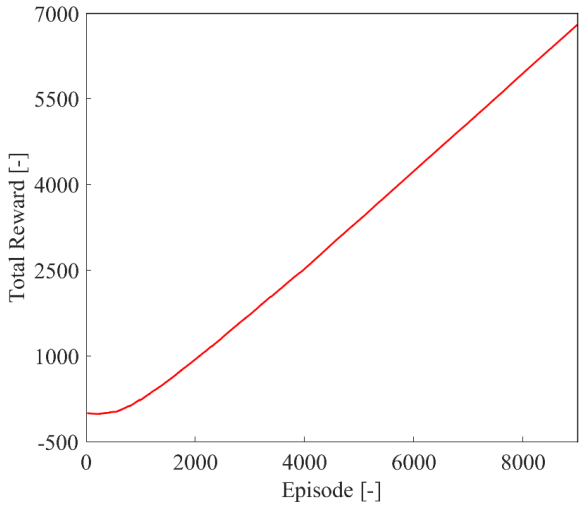


Fig. 3. A plot of the total rewards earned from the SAC networks over all of the training episodes. These results are averaged over three training trials, and a reward of one was given if both the length and twist totals fell within the randomized goal thresholds while a minor penalty was given for the mass of the arrangement at all steps.

select efficient arrangements by choosing arrangements that reach goals with the minimum amount of modules required.

V. CONCLUSIONS

Through this project, we have initiated work towards deep learning in continuous action spaces for modular robot design. We adapted a previous framework for deep RL from this project [1] and developed a plan for a hierarchical structure of neural networks that segments the process of adding a module to a current arrangement of an arm into a primary discrete section and a secondary continuous section.

We developed a potential structure for designing a modular arm arrangement where we would use a Deep-Q Network (DQN) to select either a discrete module to be added or a module group that is to be optimized for the arrangement. If a module group is selected, then the Soft Actor-Critic (SAC) algorithm would then be employed to set the continuous design variables for the type of module that is to be added to the arm.

We found that the Soft Actor-Critic (SAC) algorithm allowed us to successfully optimize continuous design variables when trained on a simplified version of our inverse kinematics tasks, and this algorithm has also led to promising results for our inverse kinematics tasks as well.

While the training of these hierarchical networks is not entirely finished, we plan on continuing to work on this project in the future. In Section 6, we go into more detail about current limitations for this work as well as our plans for this project in the future.

VI. LIMITATIONS AND FUTURE WORK

One limitation for our current work is that the SAC networks have only been trained on the length and twist variables for links. It is quite possible that different design variables that possess different action limits will require different amounts of training, and training our SAC networks on more of these design variables could help us fine tune our training. For instance, design variables could potentially be introduced to our brackets to allow for another module group that can be implemented into our SAC networks.

Looking forward, we want to optimize our SAC networks to work for the inverse kinematics tasks that we utilize for our modular robots. Once these networks would be able to return optimal design variables for a single, constrained arrangement, we would then want to integrate our SAC networks into our DQN and simultaneously train the two groups of networks at the same time. Being able to train both the DQN and SAC networks would then allow us to construct modular arrangements by selecting a module or module group through our DQN and then having our SAC networks optimize the design variables for the module that is to be added.

Beyond this, we also want to incorporate target orientations into our target positions to allow for more challenging or selective tasks. Having our modular arms satisfy both position and orientation requirements allows our arms to complete more tasks than if they were to just satisfy position constraints. For even a simple pick-and-place task, the orientation of the end-effector of a gripper or robotic arm is critical when it comes to properly picking up and setting down objects. We also want to incorporate obstacles into our environments to allow our networks to be more realistic and versatile. Forcing our modular arms to satisfy position and orientation constraints while also avoiding obstacles allows for the more robust modular designs to stand out and obtain higher rewards while also penalizing the more rigid designs that cannot adapt to obstacles.

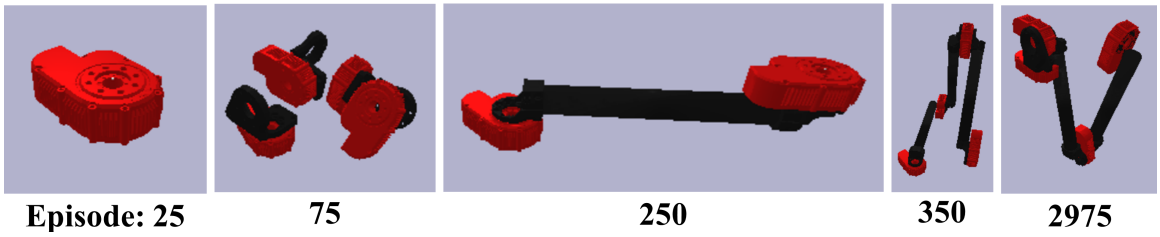


Fig. 4. Validation arrangements for the goal [0.1, 0.1, 0.1]. The numbers below the arrangements are the episodes at which these arrangements were made. Notice how at the beginning of training, the network simply outputs a single actuator which is unable to reach any positions in space. However, after only 250 episodes of training, the network is able to produce a more pragmatic design.

Fig. 5. Validation arrangements for the goal [0.2, 0.2, 0.2]. By episode 475, the network is able to produce an arrangement that can reach the desired goal position. However, by the end of the training the network has learned that only two links are needed to reach the goal instead of three.

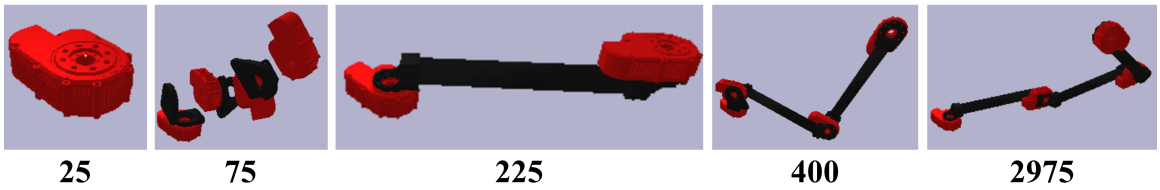


Fig. 6. Validation arrangements for the goal [0.3, 0.3, 0.3]. The arrangements for this goal actually follow a similar trajectory as those for the first validation goal. However, the arrangements for this goal end with three links instead of two for the first goal.

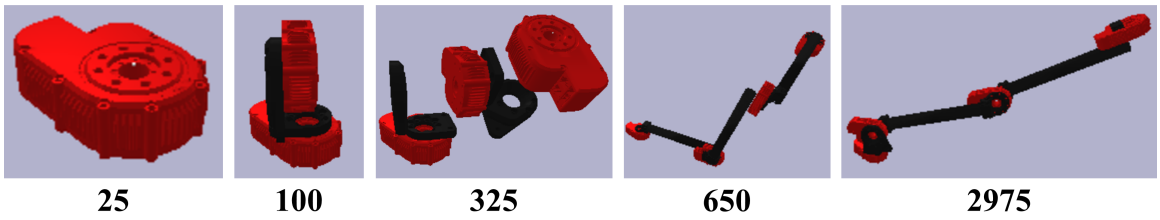


Fig. 7. Validation arrangements for the goal [0.4, 0.4, 0.4]. The first few arrangements for this goal strictly use brackets which are not very effective for reaching positions in space all by themselves. However, later on in the training the networks are able to use both links and brackets to reach the desired goal position.

Fig. 8. Validation arrangements for the goal [0.5, 0.5, 0.5]. The network appears to learn at an early point in the training that several links will be required to reach this goal position. This can be seen in the longer arrangement shown at episode 75. By the end of the training, the network still uses the same number of links as it did at episode 75, but it has moved the bracket to be earlier on in the arrangement which allows the arrangement to be more flexible.

ACKNOWLEDGMENT

I would like to thank the Biorobotics Lab for hosting me for this project and allowing me to engage with such fascinating research. I would also like to thank the Robotics Institute Summer Scholars program as well as specifically Ms. Rachel Burcin and Dr. John Dolan for working so hard

to keep this program alive for this summer. Finally, I would like to thank my home university, the University of Illinois at Urbana-Champaign, for helping me to earn the opportunity to participate in the RISS program. This material is based upon work supported by the National Science Foundation under Grant No. 1659774

REFERENCES

[1] J. Whitman, R. Bhirangi, M. Travers, and H. Choset, "Modular robot design synthesis with deep reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2020.

[2] M. Bhardwaj, S. Choudhury, and S. Scherer, "Learning heuristic search via imitation," in *In Conference on Robot Learning*, 2017, pp. 271–280.

[3] R. Desai, Y. Yuan, and S. Coros, "Computational abstractions for interactive design of robotic devices," in *In IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1196–1203.

[4] R. Desai, M. Safonova, K. Muelling, and S. Coros, "Automatic design for task-specific robotic arms," in *ICRA Workshop on Autonomous Robot Design*, 2018.

[5] E. Icer, H. A. Hassan, K. El-Ayat, and M. Althoff, "Evolutionary cost-optimal composition synthesis of modular robots considering a given task," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3562–3568.

[6] Z. M. Bi and W. J. Zhang, "Concurrent optimal design of modular robotic configuration," *Journal of Robotic Systems*, vol. 18, no. 2, pp. 77–87, 2001. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-4563%28200102%2918%3A2%3C77%3A%3AAID-ROB1007%3E3.0.CO%3B2-A

[7] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," 2018.

[8] D. Ha, "Reinforcement learning for improving agent design," *Artificial Life*, vol. 25, no. 4, p. 352–365, Nov 2019. [Online]. Available: http://dx.doi.org/10.1162/artl_a_00301

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[10] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. I–387–I–395.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[12] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5174–5184, 2018.

[13] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," 2016.

[14] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015.

[15] M. Riedmiller, "Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method," in *Machine Learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 317–328.

[16] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," 2017.

[17] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2020.

[18] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[19] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1, pp. 81 – 138, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/000437029400011O

[20] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[21] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2613–2621. [Online]. Available: http://papers.nips.cc/paper/3964-double-q-learning.pdf

[22] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*, 2015, pp. 1312–1320.

[23] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," 2016.

Multi Drone Autonomous Cinematography for Unscripted Applications

Arthur F. C. Bucker[1], Rogerio Bonatti[2], Sebastian Scherer[2]

***Abstract*— Aerial cinematography is expanding the capabilities of professional and amateur film-makers. However, safely piloting a single drone while filming a moving target in the presence of obstacles is immensely taxing, often requiring multiple highly trained human operators. The complexity of this task grows exponentially in a multi-drone system since, besides challenges regarding obstacle avoidance and artistic actor framing, factors as (1) shot diversity, (2) inter-drone collision, and (3) inter-drone visibility must be considered. Current approaches to control multiple aerial cameras presents severe real-life limitations, such as depending on fully scripted scenes or predefined shot types.**

In this work, we overcome these limitations by proposing a planning system to autonomously coordinate multiple aerial cameras for partially or unscripted applications. Our approach computes sequences of desired camera views for multiple drones around a moving target, optimizing each drone's trajectories to create diverse and artistic shots in real-time. The system was validated in a photorealistic simulator in cluttered and open environments. The results show that our system is able to attend all the requirements with an extremely low computational cost.

***Index Terms*— Motion and Path Planning, Aerial Systems, Multi-Robot Systems**

I. INTRODUCTION

Flying Cameras are revolutionizing industries that require live and dynamic camera viewpoints such as entertainment, sports, and security. The use of autonomous drones to perform tasks in these areas demonstrates several advantages in terms of cost, efficiency and safety [1] [2].

However, even with intelligent autonomous systems, these vehicles are susceptible to physical and dynamic constraints (speed, acceleration). Therefore, individual flying cameras often face challenges when dealing with unpredictable motions or cluttered environments. Shot diversity is also an inherent problem for single flying cameras, as long as invariant shots might appear 'unexciting' for the final viewer. A current solution to this problem is to record multiple takes, but it can be laborious, expensive or even infeasible in some applications. These challenges are intensified in real-world scenes, such as sports or journalistic coverage, that usually don't follow a script, and therefore cannot be reenacted. For these reasons, having access to multiple camera views can significantly improve the quality of the final video.

Currently, solutions for multi drone cinematography are scarce. Existing approaches focus mainly on coordinating vehicles to follow predefined shot types [3] or pre-planned filming missions [4] [5], restricting the use of this technology to scripted applications. Another solution in this area focuses on maximizing the 2D coverage over static targets [6], partially solving the "shot diversity" problem but only in 2D and static applications. However, none of the existing systems autonomously coordinate multiple cameras in dynamic and unscripted applications.

Aiming to address this problem and empower any individual with full artistic capabilities, we propose an autonomous multi drone system to generate diverse and artistic shots in real-time. The main contributions of this work are:

- A new framework to coordinate multiple aerial vehicles around a moving actor.
- An artistic metric for multi aerial cinematographers.
- A high-level shot type planner for autonomous filming drones.

As input, our method combines a predefined artistic principle, for example given by a certain director style, with the occupancy map of the environment and a predicted trajectory of the actor. The optimization is done using a centralized greedy planner in an actor-centered discrete spherical domain for each time step of the foreseen actor's path (figure 3). For a set of optimized high-level trajectories, each drone computes its own desired trajectory by optimizing in real-time a set of cost functions using covariant gradient descent [7].

Our approach was validated in a photorealistic simulator in cluttered and open environments, with groups of 2 to 5 drones.

The results show that our system was able to successfully coordinate the drones, safely generating diverse and pleasurable shots. Furthermore, the high-level planner demanded extremely low computational cost, optimizing each trajectory in approximately $400us$.

II. PROBLEM FORMULATION

The multi-camera problem can be divided into 2 main steps: (1) Generating the videos and (2) editing them. The first one consists of a np-hard optimization problem, and because of that, an optimal path for the cameras can not be guaranteed in a reasonable time. But, assuming that quality of all possible viewpoints of the actor can be described as a *shot quality* function, and that this function is monotonic and submodular, a greedy placement of the cameras has bounded sub-optimality guarantees [8] [9]. The second part of the problem consists of choosing a shot sequence among the generated videos. To do that a few artistic principles must be taken into considerations: first, the selected videos must have a good *shot quality*, and second, the transition between

[1]Arthur F. C. Bucker is with Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brazil arthur.bucker@usp.br

[2] Rogerio Bonatti and Sebastian Scherer are with Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA {rbonatti,basti}@cs.cmu.edu

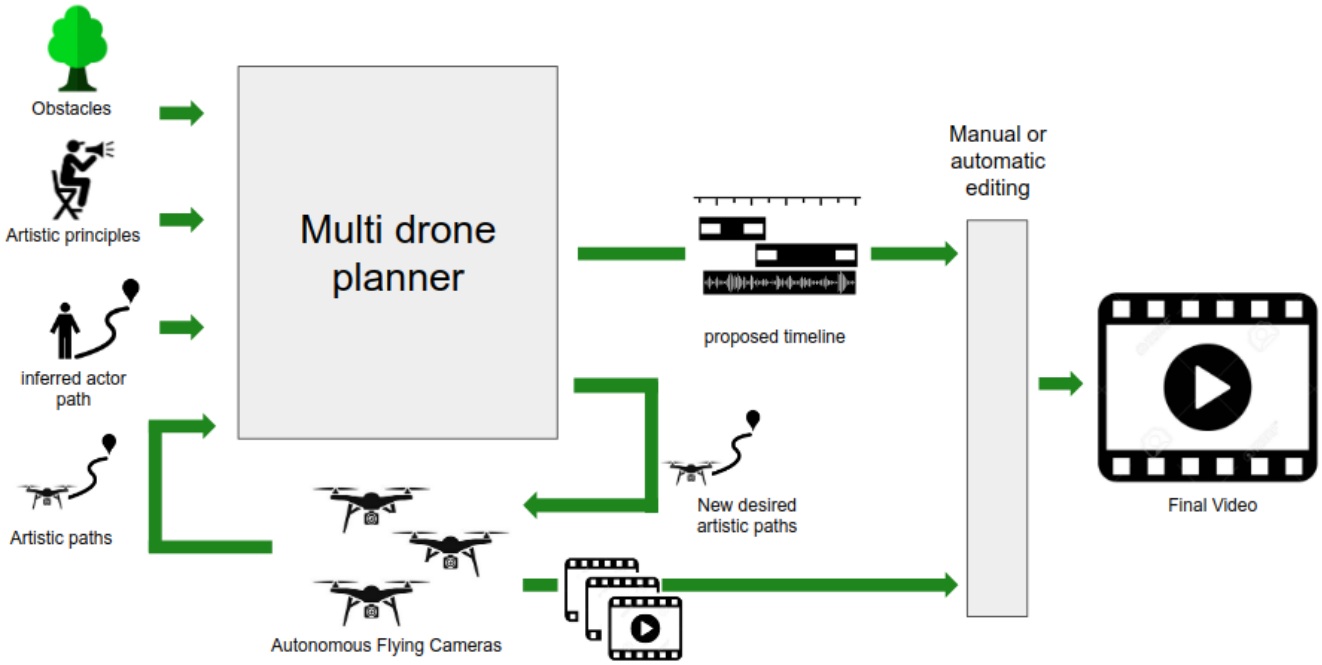


Fig. 1: Overall architecture

viewpoints must be pleasurable (e.g avoid too fast transition and also monotonous continuous viewpoints).

The *shot quality* function is qualitative and directly related to the Artistic principle followed and the spectator's taste. However, the cinematography literature presents several guidelines that can help to quantify the quality of a shot [10] [11] and the transitions between them.

Considering these artistic guidelines and safety concerns, we established a few requirements for the multi drone system:

1) Must explore shot diversity
2) Must avoid Inter-drone visibility
3) Must avoid obstacles collision and occlusion
4) Must avoid Inter-drone collision
5) Must consider artistic inputs from the user

III. RELATED WORK

A. Autonomous aerial cinematography

There is vast literature and products that address autonomous aerial cinematography. Several works and products (e.g DJI Mavic [12] and Skydio [13] [14]) focus on following user-specific artistic guidelines [7], [15]–[20], successfully controlling single flying cameras, and reducing the complexity of the filming task. However, depending on direct user input on unscripted applications can be challenging, demanding from the user continuous awareness of the scene to properly control the artistic guide-line that the drone should follow. This issue is intensified when scaling to a multi-drone system. The work [21] successfully automates this input process using deep RL, taking a step toward a fully autonomous single-drone system.

B. Multi-drone coordination

There is a rich selection of work on multi-robot systems, ranging from safety and controls, planning, target localization, exploration, and even theatrical performances. However, multi-drones solutions applied to aerial cinematography are still scarce. Current approaches address the filming task as over different optics. [3] proposes an algorithm to optimize the trajectories of multiple drones with dynamic constraints and inter-drone interactions (collision and visibility). But, similarly to the single-vehicle systems, it depends on specific user-inputted guidelines.

Another solution [5] aims to optimize a group of drones to follow a filming given mission, coordinating the vehicles considering battery and dynamic constraints. With that focusing on scripted applications.

A different approach considers the filming task as a 2D coverage problem, where shot diversity can be obtained by maximizing the visible area of the target actors [6]. Also focusing on coverage, [22] optimizes multiple flying cameras trajectories for efficiently generate datasets for 3D reconstruction.

In the context of filming outdoor events with moving targets, [4] provides an overview of the cinematography principles and existing techniques to perform filming tasks with multiple drones.

IV. APPROACH

A. Proposed solution

In this work, we propose a multi drone system capable of coordinating multiple aerial cameras around a moving actor in cluttered environments without any scripts.

The system's pipeline consists of 2 planners, the first is a centralized greedy planner that assigns high-level trajectories to the vehicles trying to optimize a handcrafted *shot quality* function. During the optimization, it also outputs a proposed timeline for real-time editing. It takes into consideration inter-drone interactions, obstacles, and artistic principles, acting as a media director. (figure 1).

The second planner, described in detail in [7], consists of an onboard planner that optimizes a single drone trajectory using covariant gradient descent and taking the high-level path as a baseline. This planner controls the fine motion of the drone in order to keep its movement smooth, ensure obstacle avoidance, and reduce occlusions.

The high-level planner preforms the optimization considering a *Artistic map* (V) that represents the *shot quality* of the possible viewpoints along the inferred actor's trajectory ($\xi_a(t)$).

B. High-Level Planner

1) Viewpoint and shot definition: Based on cinematography literature [10] [11], we select a minimal set of parameters that compose most of the possible shots for single-actor, single-camera scenarios. We define a shot Ω_{art} as a set of three parameters: $\Omega_{art} = \{\theta, \phi, r\}$, where:

- θ is the relative yaw angle between actors direction and camera ($\theta \in [0, 2\pi]$)
- ϕ l is the relative tilt angle between the actor's current height plane and the camera ($\phi \in [0, \pi/2]$)
- r is the shot scale which can be mapped to the distance between actor and camera ($r \in [r_{min}, r_{max}]$)

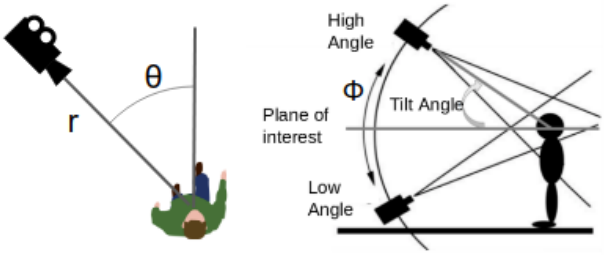


Fig. 2: Shot parameters

2) Actor trajectory definition: Let $\xi_a(t) : [0, t_f] \rightarrow R^3 \times SO(2)$ be the trajectory of the actor as a mapping from time to a position and heading (i.e., $\xi_a(t) = \{x_a(t), y_a(t), z_a(t), \theta_a(t)\}$), and let ξ_a represent the same trajectory in a finite discrete form.

3) Drone trajectory definition: We assume that our drones have a gimbal controller that can orient the camera independently of the drone's body motion. Therefore, we can purposefully decouple the drone's heading from the main motion planning algorithm. Now, let $\xi_q(t) : [0, t_f] \rightarrow R^3$ be the trajectory of the drone as a mapping from continuous-time to a position (i.e., $\xi_q(t) = \{x_q(t), y_q(t), z_q(t)\}$), and let ξ_q represent the same trajectory in a finite discrete form.

for a given drone i, let its sequence of shots be $\Omega_{qi}(t) : [0 : t_f] \rightarrow \Omega_{art}$ (i.e. $\Omega_{qi}(t) = \theta_{rel}(t), \phi(t), r(t)$). Then, the desired artistic trajectory ξ_{shots} of the drone can be defined as:

$$\xi_{shots}(t) = T(\Omega_q i(t)) \tag{1}$$

Where $T : \Omega_{art}[0, t_f] \rightarrow \mathbb{R}^3$, such that for a given actor's path ξ_a and shot $\Omega = \{\theta, \phi, r\}$, $T(\Omega, t)$ is defined as:

$$T(\Omega, t) = \xi_a(t) + r(t) \begin{bmatrix} cos(\theta_a(t) + \theta_{rel}(t))sin(\phi(t)) \\ sin(\theta_a(t) + \theta_{rel}(t))cos(\phi(t)) \\ cos(\phi) \end{bmatrix} \tag{2}$$

4) state space definition: Let A be the state space of possible shots Ω_{art}. Such as A is a half spherical discrete domain centered and oriented on each point of the foreseen actor's path ξ_a, i.e. $A = \{\Omega, t\} = \{\theta_{rel}, \phi, r, t\}$ (figure 3). Where $n_\theta, n_\phi, n_r, n_t \in \mathbb{N}_{>0}$ are the discretization factors of each degree of freedom.

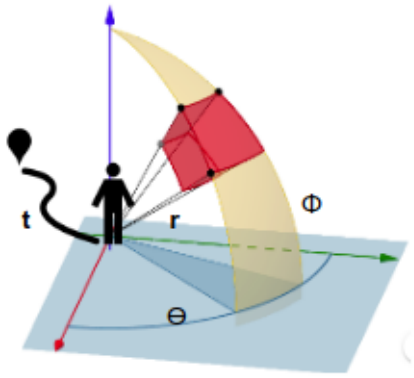


Fig. 3: State space - discrete spherical domain centered on each position of the actor's future path

We tested few sets of discretization factors and achieved satisfactory results with the following values:

$n_\theta = 16$	$n_\phi = 6$	$n_r = 6$	$n_t = 5$

With this coordinate system, is possible to drastically reduce the complexity of the planning problem.

V. Shot Quality Cost Function

Let $V : A \rightarrow [0, 1]$ be the shot quality function that maps a given shot and time to an *artistic value*. V applied in every region of A denotes what we defined as the Artistic map. The high-level planner uses this Artistic map to optimize the camera's trajectories.

Considering the requirements for the system, we define V a linear combination of 6 costs:

- Shot diversity(J_d)
- Inter-drone collision (J_{ic})
- Inter-drone visibility one to others(J_{iv1})
- Inter-drone visibility others to one(J_{iv2})
- Artistic principle prior(J_p)

- occupancy and occlusion(J_{occ})

$$V = \begin{bmatrix} w_d & w_{ic} & w_{iv1} & w_{iv2} & w_p & w_{occ} \end{bmatrix} \begin{bmatrix} J_d \\ J_{ic} \\ J_{iv1} \\ J_{iv2} \\ J_p \\ J_{occ} \end{bmatrix} \tag{3}$$

The weights of this combination are fully dependent on the application. For example, in journalistic coverage, it is probably more important to avoid inter-drone visibility than if the final application is to generate films for 3d reconstruction of the actor.

In the following section, we are going to define each component of this cost.

A. Shot diversity cost

The shot diversity cost $J_d : A \rightarrow [0, 1]$ was defined as a function of the absolute distance d of a point p in space to the position of each of the drones that have already been optimized (index $i \in [1, n_{optimized}]$).

$$J_d(t) = \sum_{i=1}^{n_{optimized}} J'_d(p, \xi_{qi}(t)) \tag{4}$$

Where:

$$J'_d(p_1, p_2) = \begin{cases} 1 & \text{if } d < d_{min} \\ d/(d_{max} - d_{min}) & \text{if } d_{min} < d < d_{max} \\ 0 & \text{if } d > d_{max} \end{cases} \tag{5}$$

with: $d = \sqrt{(p_{1x} - p_{2x})^2 + (p_{1y} - p_{2y})^2 + (p_{1z} - p_{2z})^2}$

This cost prevents the planner to allocate the drones too close to each other.

B. inter-drone collision cost

Similarly to J_d, the inter-drone collision cost was defined as a function of the absolute distance d of a point in the space and all the drones that have already been optimized. Where

$$J_{ic} = \sum_{i=0}^{n_{optimized}} J_{ic}(p, \xi_{qi}(t)) \tag{6}$$

Where:

$$J_{ic}(p_1, p_2) = \begin{cases} 1 & \text{if } d < d_{min} \\ 0 & \text{if } d > d_{min} \end{cases} \tag{7}$$

The distinction between J_d and J_{ic} is necessary to allow the control of both requirements independently.

C. Inter-drone visibility

The inter-drone visibility cost can be defined by 2 distinct components, (1) that checks if a given point p in the space would visible by a drone that was already optimized, and (2) that checks if a drone allocated in p would be able to see any of the other drones that were already optimized.

$$J_{iv1} = \sum_{i=0}^{n_{optimized}} J_{iv}(p, \xi_{qi}(t)) \tag{8}$$

$$J_{iv2} = \sum_{i=0}^{n_{optimized}} J_{iv}(\xi_{qi}(t), p) \tag{9}$$

$$J_{iv}(p_1, p_2) = \begin{cases} 1 & \text{if } p_1 \text{ is visible by } p_2 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Considering the approximation of the camera's field of view to a cone with the top angle equals to the diagonal FOV of the camera. A given point p_1 in the space would be visible by a shot of the actor at time t from p_2 if

$$\angle(\overrightarrow{(p_2, \xi_a(t))}, \overrightarrow{(p_2, p_1)}) \leq FOV \tag{11}$$

Assuming that the gimbal controller of the cameras keeps the actor centered in frame, then it is possible to say that the inter-drone visibility depends entirely on the position of the drones and the camera's FOV.

Because of that, given a discrete state-space A, the inter-drone visibility relation between 2 points can be precomputed and stored in a matrix of $m \times m$ (where $m = n_\theta \cdot n_\phi \cdot n_r$). With that reducing the computational time demanded by each optimization cycle.

D. Occupancy and occlusion cost

Because of safety and artistic reasons, it is desirable that the artistic path of the drone should not be assigned inside or behind obstacles. Therefore, given an occupancy grid (G) of the environment, the occupancy and occlusion cost J_{occ} of a shot Ω ($\Omega = \theta_{rel}, \phi, r$) at the time t was defined as follows:

$$J_{occ}(t) = \int_{r_{min}}^{r} G(T(\Omega, t))d_r \tag{12}$$

Where $G : R^3 \rightarrow [0, 1]$ is a function that maps a point in space with the occupancy of the environment at that point (0 = free, 1 = occupied)

computed in the discrete domain A:

$$J_{occ}(t) = \sum_{i=0}^{r} G'(\Omega_i, t) \tag{13}$$

With: $i \in \mathbb{N}_{>0}$ such that $\Omega_0 = \{\Theta_{rel}, \phi, r_{min}\}$ and $\Omega_{n_r} = \{\Theta_{rel}, \phi, r_{max}\}$.

and

$G' : A \rightarrow [0, 1]$ is the function that maps a region of the space c with its relative volumetric occupancy (0 = 0% occupied and 1=100% occupied). The region c is a section of A centered in $T(\Omega, t)$ as represented by the red cell in the figure 3.

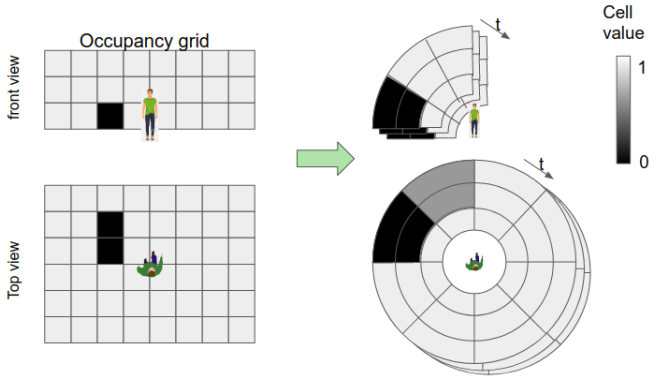


Fig. 4: Occupancy and occlusion transform (G') from a Cartesian occupancy grid to the artistic map domain A

E. Artistic principle cost (prior)

The artistic principle was defined as a user-inputted cost to penalize or give preference to specific shot types. For example, the figure 5 represents an artistic principle that penalizes shots straight from the top. With that, the planner will avoid allocating drones in these regions.

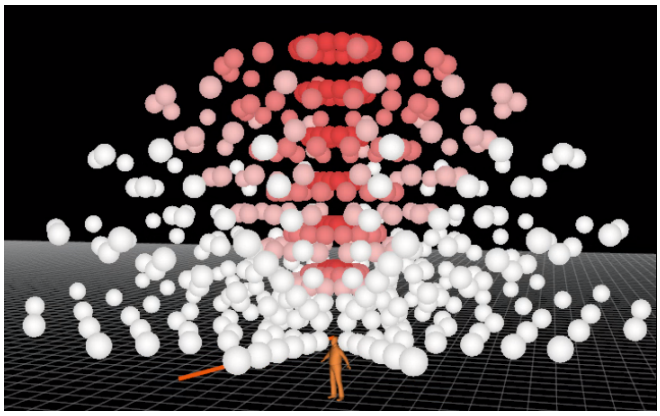


Fig. 5: Example of a artistic principle (prior) that penalizes shots straight from the top, red spheres represents regions with low artistic value

VI. Greedy planner

The high-level planner performs greedy the optimization one drone at a time based on a completely known Artistic Map (defined by the shot quality cost V applied at all points of the domain A).

Initially, for the first drone, the Artistic map is computed based only on the occupancy, occlusion, and Artistic principles. Then, for the next vehicles, the artistic map is updated with the inter-drone costs. (figure 6)

In order to compute the optimal path for each drone given an updated Artistic map, the planner computes the cumulative shot quality cost V^* of each shot (Ω) and time step, considering the next best shot (Ω') under the dynamic constraints of the drone.

$$V^*(\Omega, t_i) = V(\omega, t_i) + min(V^*(\Omega', t_{i+1})) \qquad (14)$$

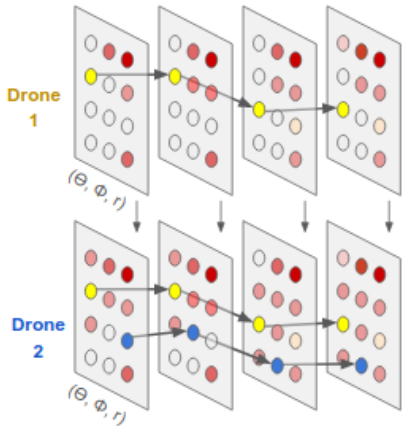


Fig. 6: Greedy planner over the Artistic map

Where, assuming the dynamic constraint that allows the drones to move only on position in the artistic domain A at a time. Then for a given shot $\Omega_{ijk} = \{\theta_i, \phi_j, r_k\}$, Ω' is:

$$\Omega'_{ijk} = \begin{bmatrix} \{\theta_i, \phi_j, r_k\} \\ \{\theta_i, \phi_j, r_{k+1}\} \\ \{\theta_i, \phi_{j+1}, r_{k+1}\} \\ \{\theta_{i+1}, \phi_j, r_{k+1}\} \\ \{\theta_{i+1}, \phi_{j+1}, r_k\} \\ \{\theta_{i+1}, \phi_{j+1}, r_{k+1}\} \\ \{\theta_i, \phi_j, r_{k-1}\} \\ \{\theta_i, \phi_{j-1}, r_{k-1}\} \\ \{\theta_{i-1}, \phi_j, r_{k-1}\} \\ \{\theta_{i-1}, \phi_{j-1}, r_k\} \\ \{\theta_{i-1}, \phi_{j-1}, r_{k-1}\} \end{bmatrix} \qquad (15)$$

VII. Local planner

After receiving a high-level trajectory, each drone optimizes its own path onboard using covariant gradient descent. The objective function, as described in detail in [7], consists of 4 costs: smoothness, shot quality, obstacle avoidance, and occlusion avoidance.

- The first one measure smoothness as the cumulative sum of *n-th order* derivatives of the trajectory.
- The shot quality cost function is given by the distance between the current camera trajectory and the desired artistic path given by the high-level planner.
- The obstacle avoidance is computed considering a truncated signed distance (TSDT) map, that contains the distance and direction of any point to the nearest obstacle.
- The occlusion avoidance cost measures how much obstacle blockage the light rays connecting the camera's and actor's trajectories would have to go through.

A contribution of this work is to consider inter-drone collision in the local planner. To do that we treat each vehicle as a moving spherical obstacle with radius of 2 meters.

VIII. Experiments and results

The approach was implemented using ROS and tested in a photo-realistic simulator (Airsim).

The tests were conducted in 2 main environments with cluttered and open regions: a forest and a neighborhood. In both setups the simulators provided a ground true position of the animated virtual actor walking close to obstacles and performing abrupt motion changes (90º turns). The drone's states were also provided by the simulator.

We tested groups from 2 to 5 drones (with FOV of 90^o) in both environments.

The state space was defined with the follow discretization factors: $n_\theta = 16$, $n_\phi = 6$, $n_r = 6$, $n_t = 5$. Considering a time window t_f of 10s , minimal and maximum radius of $r_{min} = 2$m and $r_{max} = 5$m. The inferred actor's path was computed base only on the position, speed and heading of the actor. The artistic principle was defined as shown in the figure 5. For test purposes, we assumed the weight of the shot quality function as $w_{iv2} = w_{iv2} = 0.5$ and $w_d = w_{ic} = w_{occ} = 1$ and shot-diversity parameters $d_{min} = 1$ and $d_{max} = 5$

The figure 8 shows some shots captured simultaneously by the drones in these tests.

In order to evaluate the performance of the system, in addition to a qualitative analysis of the requirements established, we computed the average planning time to optimize each drone:

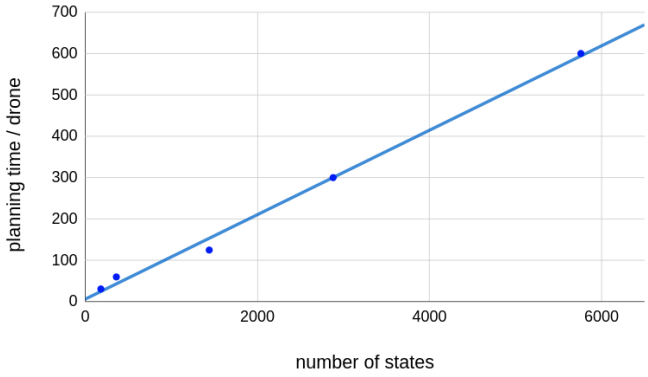


Fig. 7: Time required by the High-level planner to compute each drone's trajectory given a number of states in the artistic domain ($n_\theta \cdot n_\phi \cdot n_r \cdot n_t$) - CPU: i7-8700

The low planning time indicates that the high-level planner could easily be implemented onboard.

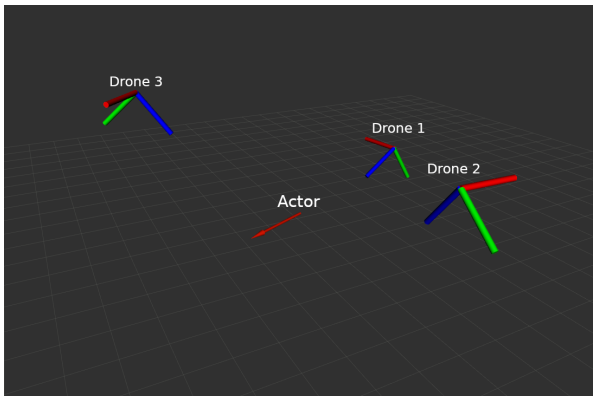
A qualitative analysis of generated scenes suggests that the system was able to fulfill all the requirements established:

1) Shot diversity was explored in all tests
2) Inter-drone visibility was avoided with no problems for the first 3 optimized drones. The fourth and fifth vehicles struggled to attend these criteria. The main reasons found to explain this behavior are: first, that there is a geometric limitation for the number of viewpoints that completely avoids inter-drone visibility; and second, due to a trade-off between shot diversity and inter-drone visibility, where higher a shot diversity weight increases the chance of inter-drone visibility.
3) Obstacles collision and occlusion were avoided, both

(a) drone 1

(b) drone 2

(c) drone 3



(d) Drone positions

Fig. 8: simultaneous shots from 3 drones in a forest environment

by the high-level trajectories and also by the trajectories followed by the drones.
4) Inter-drone collisions didn't occur in any of the tests.
5) The Artistic principle Inputted (figure 5) was followed, in a way that the top regions were only occupied by drones for less than 3 seconds in cluttered regions of the environments.

IX. FUTURE WORKS

This work takes a step towards a fully autonomous system for multi-camera aerial cinematography. Thought the processes, we raised a few questions about possible improvements in this area.

The first of them is about the viability of such a system in real-life scenarios. Our previous works [2], [7], [23] successfully evaluated single-drone systems using a similar onboard pipeline as the one here described. Showing that the setup is capable of following desired artistic paths in cluttered environments and with that establishing a solid base for the real-life validation of our multi-drone solution.

The low computing time demanded by the planner is a strong indication that the system could be implemented fully onboard and also that more exhaustive approaches could be tested. A future step for this research is to compare the greedy planner's performance against a distributed approach and a centralized optimal approach.

Another relevant question to be answered regards the artistic improvement of such a multi-camera system in diverse applications. Such a question is directly related to the editing step of the generated videos. Because of that, one future step for this work is to implement different editing algorithms and conduct user studies to quantify the artistic performance of this autonomous system. Possible comparison baselines might be videos generated with single-drone systems, human-controlled vehicles, and random shot selections.

One last point that deserves attention is the possibility of expanding the artistic capabilities of the system. Either from extracting artistic principles from existing drone-filmed scenes or though image-based algorithms to learn preferred shot types [21].

ACKNOWLEDGMENT

This work was supported by Carnegie Mellon University (CMU) RISS Program, CMU Airlab, Grupo Turing - University of São Paulo (USP), and Grupo Skyrats - USP

REFERENCES

[1] M. De-Miguel-Molina, *Ethics and Civil Drones: European Policies and Proposals for the Industry*, 2018.
[2] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *Journal of Field Robotics*, vol. 37, no. 4, pp. 606–641, Jan. 2020. [Online]. Available: https://doi.org/10.1002/rob.21931
[3] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Transactions on Graphics*, 2017.
[4] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-uav cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, 2019.
[5] L.-E. Caraballo, Ángel Montes-Romero, J.-M. Díaz-Báñez, J. Capitán, A. Torres-González, and A. Ollero, "Autonomous planning for multiple aerial cinematographers," 2020.
[6] A. Saeed, A. Abdelkader, M. Khan, A. Neishaboori, K. A. Harras, and A. M. S. Mohamed, "On realistic target coverage by autonomous drones," *ACM Transactions on Sensor Networks*, 2019.
[7] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," 2018.
[8] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.
[9] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.
[10] D. Arijon, *Grammar of the film language*, 1976.
[11] C. J. Bowen and R. Thompson, *Grammar of the Shot*, 2013.
[12] "Dji mavic," https://www.dji.com/br/mavic, August 2020.
[13] "Skydio," https://www.skydio.com/technology, August 2020.
[14] A. Bachrach, A. Bry, and M. Donahoe, "Magic wand interface and other user interaction paradigms for a flying digital assistant," June 13 2017, US Patent 9,678,506.
[15] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated cinematography with unmanned aerial vehicles," *arXiv preprint arXiv:1712.04353*, 2017.
[16] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F. o.-l. Tariolle, and P. Guillotel, "Directing cinematographic drones," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 3, pp. 1–18, 2018.
[17] N. Joubert, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, P. Hanrahan, *et al.*, "Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles," *arXiv preprint arXiv:1610.01691*, 2016.
[18] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–10, 2017.
[19] C. Gebhardt, B. Hepp, T. Nägeli, S. Stevšić, and O. Hilliges, "Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 2508–2519.
[20] M. Roberts and P. Hanrahan, "Generating dynamically feasible trajectories for quadrotor cameras," *ACM Transactions on Graphics (SIGGRAPH 2016)*, vol. 35, no. 4, 2016.
[21] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, "Can a robot become a movie director? learning artistic principles for aerial cinematography," *arXiv preprint arXiv:1904.02579*, 2019.
[22] X. Zheng, F. Wang, and Z. Li, "A multi-uav cooperative route planning methodology for 3d fine-resolution building model reconstruction," *ISPRS journal of photogrammetry and remote sensing*, vol. 146, pp. 483–494, 2018.
[23] R. Bonatti, C. Ho, W. Wang, S. Choudhury, and S. Scherer, "Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments," 2019.
[24] N. Joubert, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, P. Hanrahan, *et al.*, "Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles," *arXiv preprint arXiv:1610.01691*, 2016.
[25] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan, "An interactive tool for designing quadrotor camera shots," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–11, 2015.
[26] W. Luo, N. Chakraborty, and K. Sycara, "Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 148–154.
[27] W. Luo and A. Kapoor, "Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates," *arXiv preprint arXiv:1912.09957*, 2019.

Few-shot Learning for Interesting Scene Prediction

Yaqian Chen[1], Chen Wang[2], Yuheng Qiu[2], and Sebastian Scherer[2]

***Abstract*— Interesting scene prediction is currently under explored but is crucial for many practical applications such as autonomous exploration and decision making. For example, the subterranean autonomous exploration robots may find it interesting to have a hole in the wall; the aerial robots exploring in underground mines may change its future trajectory planning due to the discovery of a door. However, it is extremely hard for robots to decide which scenes are more likely to be interesting without human supervision. In this project, we deliver a few-shot classifier using VGG16 as a backbone and NetVLAD as data encoder. The majority voting algorithm is used to remove the annotation outliers/errors. It is demonstrated that our approach is able to find interesting scenes for practical exploration tasks in different environments like urban areas and subterranean environments. It also increases the average prediction accuracy by 13% compared to the state-of-the-art classifiers including the prototypical and matching network on very challenging robotic interestingness prediction datasets.**

***Index Terms*— Interesting Scene Prediction, Few-shot Learning, Triplet Loss, and VLAD Network**

I. INTRODUCTION

Interesting scene prediction is crucial for autonomous exploration, which is one of the most fundamental capabilities of mobile robots [1]. For example, rescue robots may need to explore an unknown environment and return all the images containing mankind to assist the rescue team. Autonomous driving systems may need to slow down the car when there is a close pedestrian or a speed limit sign. However, prior classification algorithms often have difficulties when they are deployed to unknown environments.

Most of the state-of-art classifiers fail the task due to four reasons: limited training dataset, undetermined interesting objects, expectation of working under different given environments and huge gap among interesting images. First, the number of training dataset is limited. Exploration robots are often required to predict interesting scenes online. For an online prediction, users are only able to give a few labeled images to tell their preferences. However, most of the prior classification algorithms require a large amount of training images. Second, interesting objects are not determined. For example, for underground autonomous exploration robots, some users may find bulldozers interesting. However, other users may not be quite interested in vehicles and pay more attention to scenes with construction workers. Most of previous detection systems like autonomous driving detection systems are very mature and often accomplish the accuracy near a hundred percent. However, they are only able to detect certain objects given by system builders in advance like cars or pedestrians [2]–[4]. Third, in this project we expect our system to be robust enough and will work under different environments. Take autonomous driving detection systems again for an example. It may achieve high accuracy in urban areas. However, it may not work underwater. Last but not least, the huge gap among positive images may sometimes confuse the classifiers. For example, an image with a construction worker may be considered as interesting and a photo containing an electric box may also be interesting. However, two images are far from similar, most of the classification algorithms will not classify them into the same group.

To this end, instead of detecting objects that users may be interested in, we choose to classify the robots' input images by comparing the distance between each input and each training image. We propose to build a few-shot classifier using VGG16 as backbone and VLAD network as encoder. Majority voting algorithm is used to remove the annotation outliers.

In summary, our contributions are:

- We propose a classification algorithm which is able to assist the interestingness prediction system to detect interesting scenes according to users' preference.
- We extend the state-of-art classifiers to be robust enough to work under different unknown environments.
- We propose a classification algorithm to do the few-shot classification which achieves higher accuracy on challenging robotic interestingness prediction datasets than the state-of-art few-shot learning algorithms.

II. RELATED WORK

Few-shot learning refers to the practice of feeding a learning model with a very small amount of training data, contrary to the normal practice of using a large amount of data. While there have been many improvements in designing better few-shot classifier, few works are designed for interesting scene prediction under different environments. There are mainly four different approaches to handle the few-shot learning problems, namely, model-based [5], [6], [6]–[10], optimization-based [11], [12], Bayes-algorithm-based [13], [14] and metric-based algorithms [15]–[19].

The general idea of model-based algorithms is to design the model structure to quickly update the parameters on a small number of samples and directly establish the mapping function between the input x and the predicted value P. In 2016, the Santoro's group proposed a way to solve the

[1]The author is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. email: yaqianchen@link.cuhk.edu.cn

[2]The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. email: chenwang@dr.com; {yuhengq, basti}@andrew.cmu.edu

few-shot learning task using memory enhancement algorithm based on Long Short Term Memory (LSTM) networks [5].

Meanwhile, some people paid more attention to train a meta-learner to predict the parameters of the task-specific classifiers. In 2016, Ravi and his group studied on why the gradient-based optimization algorithms fail under a small amount of data and brought out a new algorithm with an update function or update rule of model parameters [11]. The system learns a specific model in each episode instead of learning a single model in multiple episodes to enable the training model to converge. Building on the success of previous few-shot learning systems, in 2017, Munkhdalai [20] brought out the Meta Network which consists of a meta learner and a base learner. In his work, meta learner is used to learn the generalized information among meta tasks and save this information through memory mechanism. The base learner is used to quickly adapt to new tasks and interact with the meta learner to generate prediction output.

The Bayes-algorithm-based approach has relatively little connection with the above two approaches, the models are often trained by simple maximization of an evidence lower-bound using stochastic back propagation. Since variational Bayes Neural Networks (BNN) are able to approximately model the posterior over weights of a neural network, they are a natural fit for estimating the posterior over weights of a neural network that is being trained for few-shot learning [14].

The metric-based approach has been frequently used in few-shot learning area recently and most of them achieved high accuracy on different testing datasets [15] [16] [17]. The metric-based models learn one task in variant metric for all the tasks. Its basic idea is to do classification by constructing the distance between the support set and the test set. Most of the metric-based few-shot learning networks like prototypical network and matching network are composed of two parts: an encoder to extract the image feature and a classifier to calculate the distances between query dataset and support dataset.

Since metric-based approach is currently the most frequently used one in few-shot learning problems, in this project we chose to use the metric-based approach and developed an encoder and a classifier for our model.

III. METHOD OVERVIEW

As mentioned above, due to the four difficulties, namely, limited training datasets, undetermined interesting objects, expectation of working under different given environments and huge gap among interesting images, instead of using the detection models, we choose to classify the robots' input images by comparing the distances between query dataset and support dataset.

Building on current metric-based few-shot learning systems, our model mainly has three modules: clustering, training and testing. First, use the K-means algorithm to cluster the images to initialize the input image data and centroids for VLAD layer. Then, in the training process, we put the initialized images into the VLAD network and move pictures belonging to the same class closer through back propagation. Last, put in the testing dataset and get the predicted classes simply using the majority voting algorithm.

A. Clustering

For the clustering process. We use VGG16 as a backbone to encode the input raw data. Then use the K-means clustering to group image data. The output of the clustering process is regarded as the initial training data. The class centroids given by clustering will be the initial parameters for VLAD layer.

B. Training

In the training process, the system will sequentially pick one image from the training dataset to be the query dataset. Then randomly choose one image from the class of query dataset and three images from the other class as support dataset. Encode both query and support images through VLAD network. The VLAD network used in the training process consists of two parts, VGG16 and an extra VLAD layer. As the main component of VLAD network, VLAD layer is inspired by the "Vector of Locally Aggregated Descriptors" image representation commonly used in image retrieval and image classification. We also choose to use triplet loss function which separate pictures belonging to different classes and move pictures belonging to the same class closer. The detailed VLAD and VGG16 architecture as well as the formula for triplet loss function will be later introduced in Section 4.

C. Testing

In testing process, we first use the VLAD network to encode input testing image and training dataset. Through comparing the Euclidean distance between each encoded training image and the input testing data, we will find the top 5 images which are closest to the input test image. Comparing the proportion of candidate images in each class with the fixed threshold, we get the prediction.

IV. DEEP ARCHITECTURE FOR ENCODING

According to current metric-based few-shot learning systems, most works are composed of two parts: an encoder to extract the image feature and a classifier to calculate the distances between query dataset and support dataset. In this project we choose NetVLAD [21] as our encoder due to its significant robustness to translation and partial occlusion. VLAD network is based on extracting local descriptors, which are then pooled in an orderless manner. Its robustness to lighting and viewpoint changes is provided by descriptors themselves, and scale invariance is ensured through extracting descriptors at multiple scales.

VLAD network consists two parts: a dense descriptor extractor and a VLAD layer. The dense descriptor extractor is obtained by cropping the VGG16 at the last convolutional layer. Namely, the output of the last convolutional layer is a $H \times W \times D$ map which can be considered as a set of D-dimensional descriptors extracted at $H \times W$ spatial locations.

The Vector of Locally Aggregated Descriptor (VLAD) is considered as a pooling layer that pools extracted descriptors into a fixed image representation and its parameters are learnable via back-propagation. The detailed VGG16 architecture and VLAD layer are in the next sections.

A. VGG16

Visual Geometry Group 16 (VGG16) is a convolutional neutral network with 16 layers whose architecture is shown in Fig. 1. It is proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [22]. The model achieves 92.7 percent top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It makes the improvement over AlexNet [23] by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's. Since VGG16 is often more robust in classification tasks and achieves better accuracy than AlexNet in most of the cases, we choose VGG16 as our encoder backbone in our project.

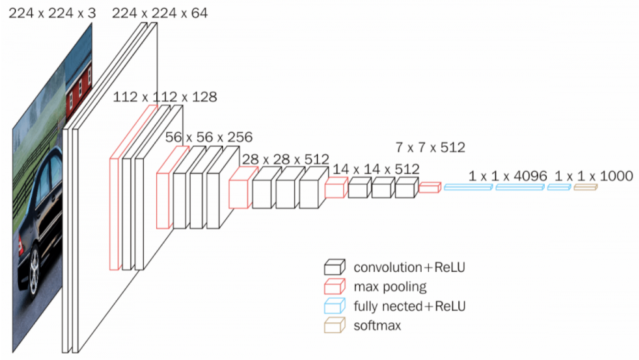


Fig. 1: VGG16 architecture

B. VLAD layer

Vector of Locally Aggregated Descriptors (VLAD) is a popular descriptor pooling method for both instance level retrieval and image classification. It captures information about the statistics of local descriptors aggregated over the image. Whereas bag-of-visual-words aggregation keeps counts of visual words, VLAD stores the sum of residuals (difference vector between the descriptor and its corresponding cluster centre) for each visual word.

VLAD layer is often regarded as a pooling layer. Given N D-dimensional local image descriptors x_i as input, and K cluster centres c_k, the layer will output the feature vector V for the input image. V is originally a $K \times D$ matrix, which is then flattened into a vector. The formula to calculate the matrix element of V in j_th row and i_th column is shown in the eq. (1). where $x_i(j)$ and $c_k(j)$ are the j_{th} dimensions of the i_{th} descriptor and k_{th} cluster centre, respectively.

$$V(j,k) = \sum_{i=1}^{N} a_k(x_i)(x_i(j) - c_k(j)), \tag{1}$$

$a_k(x_i)$ denotes the soft assignment of descriptors to multiple clusters which is defined the eq. (2).

$$\overline{a}_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_k^T x_i + b_k}}, \tag{2}$$

$\overline{a}_k(x_i)$ assigns the weight of descriptor x_i to cluster c_k proportional to their proximity, but relative to proximities to other cluster centres. It ranges between 0 and 1, with the highest weight assigned to the closest cluster centre. α is a parameter (positive constant) that controls the decay of the response with the magnitude of the distance. To further simplify the above formula, we set vector $w_k = 2\alpha c_k$ and scalar $b_k = -\alpha||c_k||^2$ and get the another form of $\overline{a}_k(x_i)$ shown in eq. (3).

$$\overline{a}_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_k^T x_i + b_k}}, \tag{3}$$

Then plugging the soft-assignment eq.(3) into the VLAD descriptor eq.(1), we get the final formula for matrix V:

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_k^T x_i + b_k}} (x_i(j) - c_k(j)), \tag{4}$$

where w_k, b_k and c_k are sets of trainable parameters for each cluster k.

V. TRIPLET LOSS

The choice of loss function is significant for classifications. Building on the current successful metric-based few-shot classifiers, triplet loss function is chosen in this project since, like Siamese networks [17], our model is also sensitive to calibration in the sense that the motion of similarity vs dissimilarity requires context. A scene is deemed interesting as long as it is more similar to the images labelled interesting compared with the images labelled uninteresting.

A triplet loss function is comprised of 3 instances with shared parameters. Input with three samples, denoted as x, x^+ and x^- where x^+ belongs to the same class as x and x^- belongs to the other, the function will output the relative similarity between x and x^+ which is defined in eq. (5):

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_k^T x_i + b_k}} (x_i(j) - c_k(j)), \tag{5}$$

where d is the cosine similarity between the output flattened descriptors. The margin m is a hyper parameter which enforces a minimum margin between the $d^2(x, x^+)$ and $d^2(x, x^-)$ and is set to 0.5.

Note that, for each input query dataset x, the loss function L is zero if the distance between the query x and x^+ is greater by a margin than the distance between x and x^-. On the other hand, if the distance between x and x^- is great than the distance between x and x^-, the result of the loss function is proportional to the amount of violation, namely, $d^2(x, x^+) - d^2(x, x^+)$.

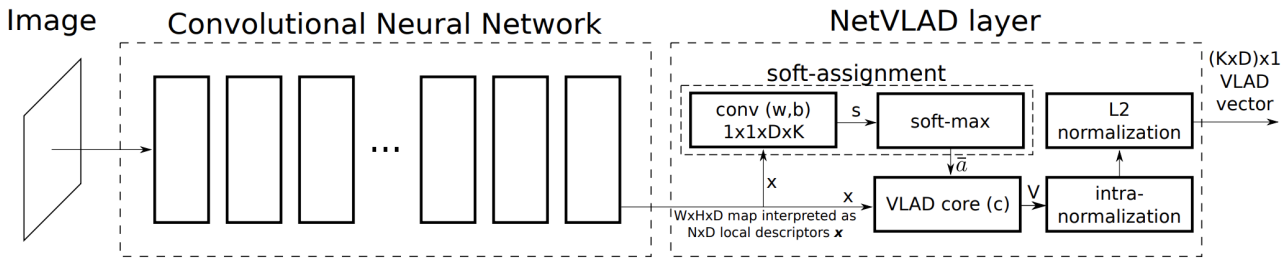


Fig. 2: NetVLAD architecture

VI. EXPERIMENTS

A. Dataset

To test the online performance on robotic systems for visual interesting scene prediction, we choose two datasets recorded by fully autonomous robots, i.e., the SubT dataset for unmanned ground vehicles (UGV) and the Cityscapes Dataset for autonomous driving system.

B. SubT Dataset

The SubT [24] is a very challenging dataset for UGV, which is based on the DARPA Subterranean Challenge (SubT) and is pre-processed via an unsupervised interesting scene prediction system [1]. The dataset environments pose significant challenges, including a lack of lighting, lack of GPS and wireless communication, dripping water, thick smoke, and cluttered or irregularly shaped environments. Each of the tunnels has a cumulative linear distance of 4-8 km. The dataset is generated from seven long videos taken by two unmanned ground vehicles (UGV), the detail information for seven videos and the dataset generated from each sequence is in table 1. Part of the interesting and uninteresting scenes taken from the dataset are showed in Fig. 3. The left two columns are the training images labelled as uninteresting and the right three columns are labelled as interesting. Each sequence is evaluated by at least 3 persons.

Fig. 3: Interesting and Uninteresting Scenes in SubT Dataset

C. Cityscapes Dataset

The Cityscapes Dataset [25] focuses on semantic understanding of urban street scenes with totally 30 classes. The dataset is challenging for the classification task, collecting images from fifty different countries and under various weather conditions. The image labelling is based on the principle that only images with close enough (with distances no more than three meters) vehicles and pedestrians are labelled as interesting. The dataset is evaluated by two people and only 22.8 percent of the images are labeled as interesting by both persons. Some interesting and uninteresting scenes taken from the Cityscapes dataset are displyed in Fig. 4. The left two columns are the training images labelled as uninteresting and the right three columns are labelled as interesting.

Fig. 4: Interesting and Uninteresting Scenes in Cityscapes Dataset

D. Implementation

We train the model using the Adam optimizer with a learning rate of 0.0001 and set the learning rate step to be 5. Two base architectures which are extended with VLAD layer are used in the project: AlexNet and VGG16. Both of them are cropped at the last convolutional layer before ReLU. For NetVLAD we set the number of cluster to 2 resulting in 1024 and 512-D image representations for the two base architectures, respectively. We train the parameters of the model using Stochastic Gradient Descent (SGD) on tuples (x, x^-, x^+) which are generated from training dataset. In the majority voting process we set the threshold based on the priori probability. For example, if the uninteresting images takes around 60 percent in the training dataset, the query dataset will only be deemed as uninteresting when more than 60 percent of candidates are predicted to be uninteresting.

E. Results

Baselines and state-of-the-art. In order to evaluate the performance of our model, we compare our classifier with the "off-the-shelf" few-shot learning classifiers on interesting scene prediction. Matching network [16], published in 2017, has shown an excellent performance in classification on both

TABLE I: The information of the SubT dataset.

Video	I	II	III	IV	V	VI	VII	Overall
Length(min)	53.1	55.7	79.4	80.0	59.0	57.5	83.0	467.7
Interesting(%)	31.5	56.0	37.0	26.0	27.0	53.5	16	35.2

Omniglot and miniImageNet datasets. Prototypical network [15] learns a metric space in which classification can be performed by computing distances to prototype representations of each class. The comparison result of our method vs. off-the-shelf networks is shown in Fig. 5.

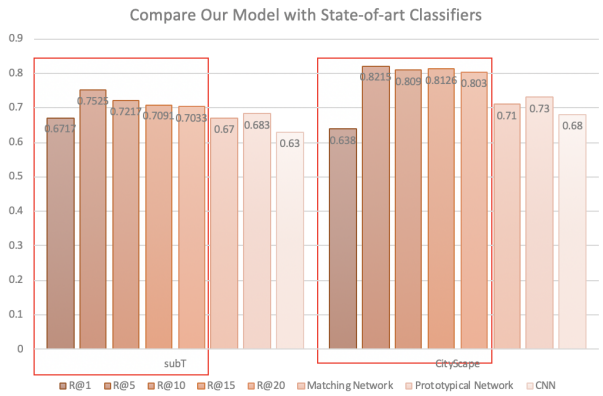


Fig. 5: **Comparison of our methods versus off-the-shelf networks:** our few-shot classifier (R@1,R@5,R@10,R@15,R@20 which are circled out and denote VLAD model using top1, top5, top10, top15 and top20 candidates for majority voting respectively) outperform by a large margin off-the-shelf ones(matching network, prototypical network and CNN)

As we can interpret from the histogram, our few-shot classifier outperform off-the-shelf ones by a large margin. The model based on NetVLAD increases the average accuracy by roughly 13 percent on SubT and by 10 percent on Cityscapes datasets. The best performance often occurs when we use the top five candidates to do the majority voting and matching network has higher prediction accuracy on these two datasets than the prototypical network.

Furthermore, we compare our model performances with different backbones, namely, Alexnet and VGG16. The best performances on both datasets are given based on VGG16. However, the results are various from dataset to dataset.

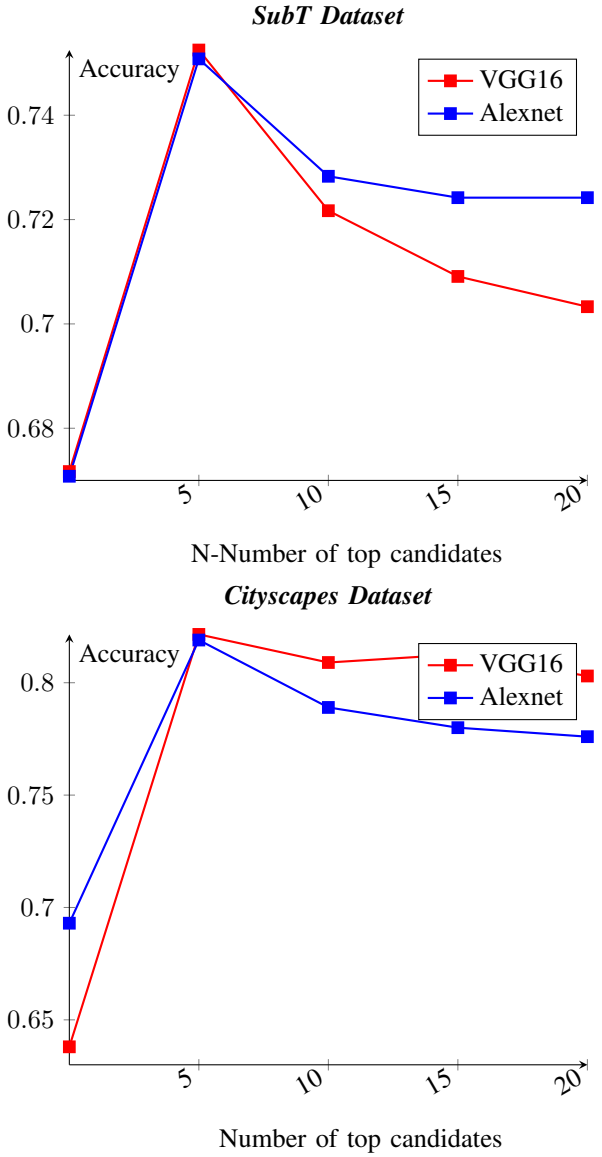


Fig. 6: Comparison of VGG16 versus Alexnet as Backbone

VII. CONCLUSION

In this project, we have designed a new few-shot classifier that is trained for interesting scene prediction in an end-to-end manner. Our model outperforms off-the-shelf few-shot learning models like prototypical network and matching network on the challenging SubT dataset, as well as on Cityscape dataset. Building on the current successful few-shot learning models, our algorithm mainly has two parts: an encoder based on NetVLAD and a classifier to perform the majority voting algorithm. In the experiment section, we proved that our model is able to obtain a relatively high accuracy under different environments.

ACKNOWLEDGMENT

The work is supported by the Air Lab at the Robotics Institute, Carnegie Mellon University and the Institute of Artificial Intelligence and Robotics for Society (AIRS), The Chinese University of Hong Kong, Shenzhen. We also give our special thanks to Rachel Burcin, Dr. John Dolan and all the members in Robostics Institute Summer Scholars (RISS) program for making this program possible.

REFERENCES

[1] C. Wang, W. Wang, Y. Qiu, Y. Hu, and S. Scherer, "Visual Memorability for Robotic Interestingness via Unsupervised Online Learning," in *European Conference on Computer Vision (ECCV)*, 2020.

[2] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," 2018.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2019.

[4] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[5] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," 2016.

[6] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," 2018.

[7] Z. Xu, L. Zhu, and Y. Yang, "Few-shot object recognition from machine-labeled web images," 2016.

[8] B. N. Oreshkin, P. Rodriguez, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," 2018.

[9] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," 2018.

[10] S. Qiao, C. Liu, W. Shen, and A. Yuille, "Few-shot image recognition by predicting parameters from activations," 2017.

[11] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017.

[13] Li Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[14] M. Jaques, *Conditional Bayesian Neural Networks for Few-Shot Learning*. Master of Science by Research Data Science University of Edinburgh, 2018.

[15] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," 2017.

[16] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," 2016.

[17] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," 2015.

[18] E. Triantafillou, R. Zemel, and R. Urtasun, "Few-shot learning through an information retrieval lens," 2017.

[19] S. Motiian, Q. Jones, S. M. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," 2017.

[20] T. Munkhdalai and H. Yu, "Meta networks," 2017.

[21] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," 2015.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[24] T. I. Murphy, "Line spacing in latex documents," [EB/OL], http://theairlab.org/dataset/interestingness Accessed April 4, 2010.

[25] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," 2016.

[26] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," 2014.

[27] K. Koreitem, F. Shkurti, T. Manderson, W.-D. Chang, J. C. G. Higuera, and G. Dudek, "One-shot informed robotic visual search in the wild," 2020.

Trajectory Planning for Self-Driving Using Reinforcement Learning

Josiah Coad[1], Zhiqian Qiao[2], John Dolan[2]

***Abstract*— Controlling autonomous vehicles on roads in complex environments is a difficult task. Complexity for self-driving comes in the form of a large state space, multiple objectives and the many other stochastic obstacles involved. Reinforcement learning has been proposed to produce controllers for self-driving cars. However, for some tasks, generating a control directly cannot guarantee the smoothness and safety in a desirable manner. Our work proposes a model-free deep reinforcement learning trajectory planner that exhibits safe, comfortable and generalizable behavior. We evaluate the performance of the ego car on overtaking behavior in dynamic urban environments. From our results we show that our method exhibits fewer crashes and less unnecessary acceleration and steering compared to RL controller methods.**

***Index Terms*— Autonomous Driving, Trajectory Planning, Motion Planning, Reinforcement Learning**

I. Introduction

Research in autonomous driving is a rapidly developing field in academia and industry because of its potential to reduce traffic accidents and congestion and increase mobility. However, controlling autonomous vehicles in a safe and comfortable manner on roads in dynamic environments is an open problem. We measure safety by lack of collisions and comfort by lack of unnecessary vehicle controls. Current approaches to vehicle control either lack in generalizability or their ability to handle dynamic environments, or they produce jerky and unsafe driving.

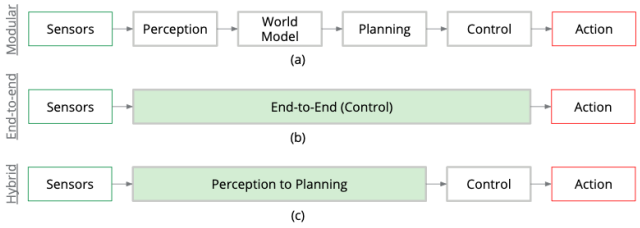


Fig. 1. Comparison of common approaches to self-driving. Green represents learning components. (a) Maps sensors (such as RGB camera) to actions (such as throttle, steer and brake) through a modular pipeline. (b) Maps sensors to actions directly by manipulating the car controls. (c) A hybrid approach which uses deep learning to map sensors to a planned trajectory and then passes the trajectory onto a controller. Figure adapted from NeuroTrajectory [1].

Traditionally, autonomous driving has been addressed through a modular pipeline of perception, route planning, behavior decision-making, trajectory planning and finally, control [2], shown in Fig. 1(a).

Traditional approaches to the trajectory planning module, such as spline or lattice planners, have been proposed. However, these systems require hand engineered policies and cost structures which limit their generalization in complex environments.

End-to-end deep learning approaches to vehicle control, as shown in Fig. 1(b), have been proposed using imitation learning (IL) [3]–[6], reinforcement learning (RL) [6]–[10] and a combination of the two, inverse reinforcement learning [11]. However, directly mapping a large state space to controls is a hard problem. End-to-end systems have a hard time ensuring functional safety — because machine learning is tasked with optimizing an expectation over many instances. In other words, these end-to-end models are generally not very interpretable, so it can be difficult to understand how a control decision was made for evaluation and debugging the system. Furthermore, using deep learning to directly control the vehicle is reactive and thus tends to be unsafe and jerky.

Because of the limitations of traditional (non-deep learning) trajectory planning in complex environments, and the safety and comfort concerns of end-to-end systems, research in recent years has turned to planning a trajectory via deep learning, as seen in Fig. 1(c). By using deep learning, this approach has the potential to behave well in environments defined by a large state space, multiple objectives and many stochastic moving obstacles. And by separating the control module, and allowing the same to be executed by a non-machine learning based controller allows for more interpretability and safety constraints [12].

Predominantly, IL has been proposed as the deep learning approach for the task of trajectory planning [1], [4], [10], [13]–[15]. IL learns to imitate the trajectories of an expert driver. Issues arise with using IL for this task, however, in handling edge cases such as near-crash scenarios, generalizing to unseen cases and the need for large amounts of labor-intensive hand-collected training data.

In this paper, we propose that instead of using IL to plan a trajectory, we use reinforcement learning. RL has an advantage over IL in that it learns by playing in a simulation. By doing so, RL explores the simulated driving environment in many more edge cases than IL and does not require any hand-collected training data. Thus, RL is better equipped to handle edge case scenarios. In addition, the ability for RL to perform better than human experts has been widely demonstrated on tasks such as playing Atari Games and controlling robots [16]. Most vehicle accidents today happen due to human driving error which begs the question if a

[1]Josiah Coad is with The Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA, josiahcoad@tamu.edu

[2]Zhiqian Qiao and John Dolan are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, {zhiqianq, john}@andrew.cmu.edu

human driver is really an *expert* driver that we want to emulate. RL could theoretically perform better than human drivers.

We propose a method to adapt a Proximal Policy Optimization model-free reinforcement learning algorithm [17] to generate a trajectory of waypoints for the ego vehicle to track with a PID controller. Our learned policy generates trajectories which are safe, comfortable, and efficient and respect traffic rules.

The structure of this paper is as follows: in Section II we present related works and how they motivate our approach. In Section III we formalize the problem and our solution. In Section IV, we explain our custom environment and how we collect our results. In Section V we demonstrate our algorithm in our simulated driving environment and show that compared to control-based RL, our approach results in fewer crashes and less acceleration and steering. Finally, in Section VI and VII we summarize our contribution and provide direction for further work.

II. Related Work

A. Non-Deep Learning Trajectory Planning

A trajectory is a sequence of local states, such as cartesian coordinates and velocity, parameterized by time, to be visited by the vehicle. Trajectory planning is the task of planning a feasible trajectory within some lookahead distance of the vehicle that does not result in a collision and moves towards a goal. An optimal trajectory should avoid obstacles such as other cars and pedestrians, adhere to a set of motion constraints and respect traffic rules [18].

Many non-deep learning approaches to trajectory planning have been proposed as part of the modular pipeline. Typical approaches involve rapidly expanding trees [19] or lattice sampling [20]. However, many of these systems make strong simplifying assumptions about the environment, such as no uncertainty in an obstacle's motions or ignoring motorcycles and non-motorized traffic participants or limiting the vehicle movement [18]. These systems also require extensive human engineering to tune. Thus, many of the traditional trajectory planners are not able to handle complex environments defined by a larger state space such as RGB images, multi-faceted objectives and many stochastically moving obstacles.

For a full review of non-deep learning methods to trajectory planning and their limitations, refer to the surveys [18], [21].

B. End-to-End Reinforcement Learning Methods

In End-to-End Race Driving with Deep Reinforcement [7], the authors train an RL agent to drive a race car around a track by directly controlling the car with the agent. They train and test their performance on a simulator with three high fidelity tracks across varying terrain. However, the ego vehicle exhibits aggressive unnecessary steering behavior and is not tested in any environment that includes other vehicles.

In the release of the CARLA simulator [10], the authors introduce a baseline end-to-end RL system that they used to compare in urban driving performance to a module pipeline approach and an end-to-end IL approach. In their results, they found the end-to-end RL control approach to have inferior performance to the other approaches.

In the recent work [9], the authors separately train the perception module to derive high level features about the world and then a world-to-control module to control the vehicle. In CIRL [6] and [9] a gating unit is used to choose which high-level behavior to follow (lane-following, left, right or straight) in urban driving. However, even in their best results, the authors still report crashes, motivating the need for future work. They do not report on the smoothness of the driving policy. In [22], the authors use a PPO RL agent to control the derivative of acceleration and steer, which is jerk and steering rate. They claim this increases the comfort of the ride by reducing oscillatory and jerky behavior.

For a complete review of end-to-end controllers, refer to the surveys [23], [24].

C. Imitation Learning-Based Trajectory Planning

In trajectory planning, the system is forced to plan ahead by outputting a sequence of waypoints, i.e. a trajectory, instead of simply choosing the next steering and acceleration commands, like in end-to-end systems. Waypoints are independent of car geometry and waypoint-based trajectories are easier to interpret and analyze than low-level network outputs such as steering commands. [24] This means we can know ahead of time where the car is planning to move and add safety constraints to the path as in [12].

The predominant deep learning-based approach to do trajectory planning is imitation learning. In [13], the authors propose a deep imitation learning trajectory planner. Here, the authors aim to learn the parameters of distribution which maximize the likelihood of an expert path conditioned on the agent's observation. They fit the imitative model in a supervised manner using a recurrent neural network. In a subsequent paper, NeuroTrajectory: A Neuroevolutionary Approach to Local State Trajectory Learning for Autonomous Vehicles [1], the authors use imitation learning and evolutionary methods to produce trajectories by learning to imitate expert drivers. Recently, Learning by Cheating [14] learned to extract a high level state space before training the imitation learning planner. ChauffeurNet [15] proposed imitation learning trajectory planning with the addition of perturbations to the experienced trajectories for the sake of robustness. [12] proposed using a feed forward neural network to generate a trajectory with an output layer of size $2H$ where H is the size of the horizon. The output represents a trajectory $[x_{t+1}, y_{t+1}, \ldots, x_{t+H}, y_{t+H}]$. Their goal is to minimize the displacement d_{t+i} between the expert's actual trajectory point (x_{t+i}, y_{t+i}) and the predicted point $(\hat{x}_{t+i}, \hat{y}_{t+i})$:

$$d_{t+i} = ((x_{t+i} - \hat{x}_{t+i})^2 + (y_{t+i} - \hat{y}_{t+i})^2)^{\frac{1}{2}}$$

Their overall loss function is defined as

$$\mathcal{L} = \frac{1}{H} \sum_{i=1}^{H} d_{t+i}^2$$

Imitation learning still faces limitations, however, in that it requires large amounts of human driver training data, which is laborious to collect. Furthermore, the learned policy tends to overfit to the environment which it was trained on and could fail if presented with a vastly different situation, such as recovering from a near-crash scenario, if this scenario was not well covered in the training data. Thus, the trained algorithm may not react appropriately in these conditions in deployment. A recent paper argues that even with tens of millions of examples, direct imitation learning sometimes does not yield satisfactory driving policies [15].

D. Inverse Reinforcement Learning For Cost Maps

Watch This [25] uses inverse reinforcement learning (IRL) to produce cost maps from camera input based on one year's worth of trajectory information. Trajectories are then produced from the cost map by taking a soft argmax. The authors demonstrate that IRL achieves better performance than manually constructed cost functions for their purpose. However, as the algorithm is still dependent upon large amounts of data, it still suffers from similar drawbacks of imitation learning as discussed in Section II-C.

E. Reinforcement Learning Trajectory Planning

We have found three works that have proposed methods similar to reinforcement learning trajectory planning, but none have done it successfully in environments involving other moving obstacles. Deep imitative models for flexible inference, planning, and control [13] proposes a model-based reinforcement learning control planner as an aside to compare with their imitation learning algorithm. They perform a breadth first search cost minimization in a reachability tree through free-space. However, the planner exhibits inferior performance compared to their other methods, especially in dynamic environments. One reason for the inferior performance is that they explicitly predict that each car in the ego vehicle's view has a constant velocity and straight-ahead direction, which is an oversimplifying assumption.

In [26], the authors propose a method for using RL for trajectory planning in static environments via a transfer model. The transfer model is used to make a world model. The RL agent then interacts with the world model by directly controlling a virtual car equipped with bicycle dynamics. The resulting trajectory of the virtual vehicle is recorded and mapped back to the real world frame. A PID controller is then used to control the real car and track the trajectory. The limitation with this is that the transfer model, which is the bottleneck for the performance of the system, does not scale well to complex environments and dynamic obstacles. Indeed, the system was only tested on simple static environments and demonstrated in real life on a single turn. Thus, the need for a RL trajectory planner that is able to handle dynamic environments is not yet satisfied.

In [27], the authors train a DDPG model to plan a trajectory for a constant-speed ego vehicle. Their method uses the current ego-vehicle state (x,y,speed,heading) and goal state as the observation. The DDPG action is two dimensional and represents two consecutive relative y values in front of the vehicle between the current state and goal state. They then fit a spline to these four points. The main objective of the training is to learn to plan a feasible trajectory. Thus, the reward structure focuses on reachability. Road dynamics and moving obstacle vehicles are not considered.

We note that non-machine learning-based trajectory planning performs well in simple environments but does not scale well. We further note that trajectory planning using imitation learning has achieved state-of-the-art results thus far but suffers from being data hungry. Finally, we see the progress that reinforcement learning has found in directly controlling a car, although so far, it has not been used successfully for trajectory planning in dynamic environments. Thus, we wish to take inspiration from all these approaches to propose a novel solution. We propose a method to use model-free reinforcement learning to perform trajectory planning in dynamic environments involving other moving obstacles.

III. Methods

The goal of trajectory planning is to find a sequence of points $\vec{\phi} = \langle \phi_1, \phi_2, ...\phi_n \rangle$ where $\phi_i = (x, y, speed, heading)$ indicates the path that the ego vehicle should travel to avoid obstacles, respect road rules and maximize comfort and efficiency towards a goal. Once we obtain $\vec{\phi}$, we can pass it on to a controller, such as a PID controller. The PID outputs a control c where $c = (steering, throttle, brake)$ which is applied directly to the actuators of the car.

Reinforcement learning seeks to learn an optimal policy π parameterized by θ which maps a state (observation) s in the observation space O to an action a in the action space A, i.e. $\pi_\theta(a|s)$. In deep RL, θ are the weights to a neural network. Unlike supervised learning, RL does not require any labelled training data. Instead, RL interacts with an environment by being presented with s_t and choosing a_t according to its policy π. Under the assumption of full observability, RL operates on the premise that the environment can be viewed as a Markov decision process (MDP) with a (possibly stochastic) transition function $T(s_{t+1}|s_t, a_t)$. The environment also returns a reward $R(s_t, a_t)$ and a binary signal, done. The process of observing s_t, getting action a_t according to the policy π, and executing a_t in the environment is called a step. The reinforcement algorithm works to find the policy parameterization that maximizes the expected reward over the episode, i.e.

$$argmax_\theta \mathbb{E}[R(s_t, a_t)|a_t \sim \pi_\theta(s_t), s_t \sim T(s_{t-1}, a_{t-1})]$$

Both continuous and discrete actions are possible with reinforcement learning, depending on the RL algorithm used. In the case of discrete, A is a (generally small) finite action space. In the case of continuous, A has a shape of $[-1, 1]^n$ where n is the number of controls contained in one action. The constraint to $[-1, 1]$ is trivial since we can scale the action once inside the step function. However, empirically

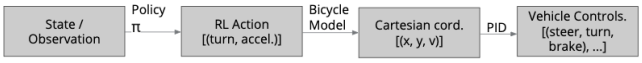


Fig. 2. Our approach to trajectory planning with reinforcement learning

this constraint is better for convergence. In the case of end-to-end or control-based RL, we may have $n = 3$ controls such as steering, throttle and brake. However, finding a way to produce a trajectory $\vec{\phi}$ from a is not trivial and is the main contribution of this paper.

A. Trajectory Generation

To convert an action produced by our policy to a trajectory, the policy chooses an $a \in [-1, 1]^{2n}$ where n is the number of points we plan in the trajectory. This action can be thought of as a sequence of n "long range control" (acceleration, turn), i.e. (α, δ) tuples, denoted $\vec{c_*}$. However, $\vec{c_*}$ is not used to directly control the car. Instead, in the planning procedure, we iterate through each $c_* := (\alpha, \delta) \in \vec{c_*}$ and use the PID controller along with a bicycle dynamics model to extrapolate what would be the resulting state $\phi = (x, y, speed, heading)$ of applying these controls for a constant "long range" time τ_*. Thus, a is converted to the trajectory $\vec{\phi}$. Then, in the control procedure, $\vec{\phi}$ is fed to the PID controller to produce "short range" controls $\vec{c}$ which gets applied to the car at "short range" time τ for each $c \in \vec{c}$. This is one step of the RL agent. Thus, in total, one step of the RL agent takes $n\tau_*$ time in the environment. Fig 2 shows this procedure, Algorithm 1 formalizes this RL-step and Algorithm 2 formalizes the planning procedure. For our experiments, we assume τ_* is a multiple of τ. Note that if the ego vehicle adheres to the bicycle model, the trajectory will be entirely reachable. Table I conveys the parameter settings we used in our implementation of Algorithm 2.

Algorithm 1 RL-Step

procedure RL-STEP
  $s \leftarrow$ observe ▷ Observe environment state
  $\phi, \cdots \leftarrow$ s ▷ Extract ego's current state
  $\vec{c_*} \leftarrow \pi$(s) ▷ Get long range controls
  $\vec{\phi} \leftarrow$ RL-Plan($\vec{c_*}, \phi$) ▷ Transform $\vec{c_*}$ to trajectory
  ▷ *Control Stage*
  for ϕ in $\vec{\phi}$ **do**
    for τ_*/τ iterations **do**
      $c \leftarrow$PID(ϕ, τ)
      drive(c, τ) ▷ Apply c to actuators for τ time
    end for
  end for
end procedure

By requiring the RL algorithm to plan multiple steps into the future, we force it to account for the dynamics of other vehicles. Thus, our agent inherently learns to plan in dynamic environments such as urban conditions.

Algorithm 2 RL-Based Trajectory Planner

procedure RL-PLAN($\vec{c_*}, \phi$)
  $\vec{\phi} \leftarrow$ empty array ▷ Initialize trajectory
  for c_* in $\vec{c_*}$ **do**
    $\phi' \leftarrow$ bicycle(c_*, ϕ, τ_*) ▷ Use bicycle model to calculate outcome of applying "long range" control
    for τ_*/τ iterations **do** ▷ Adjust for motion constraints imposed by the PID controller
      $c \leftarrow$PID(ϕ', τ)
      $\phi \leftarrow$ bicycle(c, ϕ, τ)
    end for
    $\vec{\phi}$.append(ϕ)
  end for
  return $\vec{\phi}$
end procedure

TABLE I
ALGORITHM PARAMETER SETTINGS

Parameter	Value
Waypoints (n)	2
Time Between Points (τ_*)	0.3 s
Time Between PID Controls (τ)	0.05 s

B. Reward Structure

Reinforcement learning learns a policy which maximizes a reward (i.e. minimizes a cost) over an episode. We choose a reward/cost structure for the agent which is designed to promote desirable driving behavior. We consider driving off the road or crashing into a vehicle (using a circular collision boundary around each car) a failure. Reaching the end of the road segment is considered a success. A failure or success terminates the episode. We assign a cost to the normalized distance from the center of the closest lane. To avoid unnecessary lane changes, we assign a constant cost to each lane switch. Finally, in an effort to respect road rules, we assign a coefficient cost for normalized error off the target speed. Our reward function is thus a linear combination $r(s, a) = \boldsymbol{wx}$. Refer to Table II for $\boldsymbol{w}$ and $\boldsymbol{x}$ definitions.

C. Observation Space

The agent is equipped with ray casting object detection, which we implement as k rays (in this paper, k=51) spread equally in some field of view (FOV) (in this paper, FOV=$\pm$90°) to detect distance to the closest obstacle (or some max lookahead distance, whichever is less). For graphical representation, refer to Fig. 5. The agent also knows the

TABLE II
REWARD STRUCTURE SETTINGS

Measure ($\boldsymbol{x}$)	Range	Coefficient ($\boldsymbol{w}$)
fail (crash / offroad)	{0,1}	-50
success	{0,1}	10
lane cross	{0,1}	-5
speed error	{0,1}	-0.8
off current lane center	[0,1]	-0.8
tailgating	[0,1]	-1

TABLE III

HYPERPARMETER SETTINGS FOR PPO

PPO Paper Name	SB PPO2 Name	Value
Number of actors	n_envs	32
Horizon (T)	n_steps	64
Minibatch size[1]	nminibatches	64
Discount (γ)	gamma	0.999
Adam stepsize	learning_rate	$2e^{-4}$
Entropy coefficient (c_2)	ent_coef	0.01
Clipping epsilon	cliprange	0.4
Num. Epochs	noptepochs	25
GAE parameter (λ)	lam	0.99
Policy network	net_arch vf	FC: [64, 64]
Value network	net_arch pi	FC: [64, 64]

[1] nminibatches was set to 64 in the SB PPO2 implementation. However, this equals a minibatch size of 32 since mini-batch_size = (n_envs*n_steps)/nminibatches.

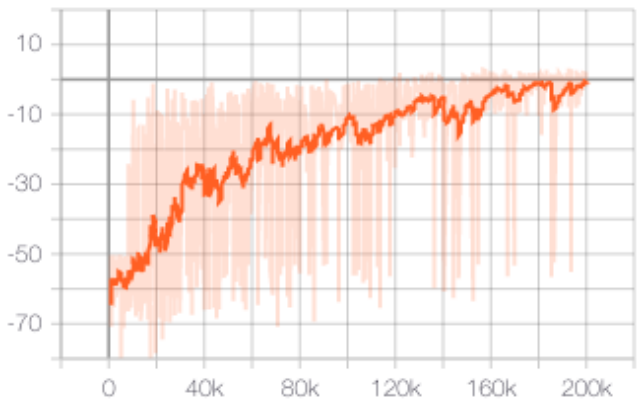


Fig. 3. Episode rewards from 200,000 steps during training. Visualization produced from tensorboard with 0.93 smoothing.

ego vehicle speed, ego target speed, ego heading and the lateral displacement from the center of the road. Currently, the other vehicles have a constant speed and direction, so the agent does not need to learn to extract this information. To generalize our approach to work with variable speeds and directions of other vehicles, we believe that using an LSTM could be a promising research direction (see Section VII for more details).

IV. EXPERIMENTS

A. Network Architecture

To represent the policy, we used a fully-connected (FC) multilayer perception network with two hidden layers of 64 units, and tanh nonlinearities, outputting the mean of a Gaussian distribution, with variable standard deviations. We don't share parameters between the policy and value function (so coefficient c_1 from the original PPO paper [17] is irrelevant).

Hyperparameter tuning was found to help PPO reach better performance. We performed hyperparameter tuning with the Python package Optuna [28]. In Table III we outline the hyperparameter settings we found to work best via a hyper-parameter search. We used the Python package called stable baselines (SB) version 2.1 [29] for their implementation of PPO called PPO2.

B. Training details

In Fig. 3 we show some of the episode rewards during training from 2M steps. The training was run on a Tesla K80 GPU with 32 parallelized environments and took about 4 hours of wall time in our custom environment.

Each episode in training was initialized with one obstacle vehicle placed randomly on the road segment and the ego vehicle placed randomly in one of the three lanes. The agent was limited to highway speed (54 to 209 kmh). See Table IV for more details on the environment settings that were used during training.

C. Custom Environment

We implemented a custom environment which contained all the components we needed for proof of concept. This allowed us to quickly run experimental iterations. The simulation environment consists of a straight-road segment with three constant width lanes. The sim. cars move with constant direction and speed. The purpose behind this setup is to give the agent adequate room to learn to switch lanes and demonstrate behaviors of overtaking. See Fig. 5 for demonstrations of our environment. Table IV reports the parameters we used in our simulated environment along with their setting (if constant) or settings (if variable). In the future, we wish to move to a more high fidelity driving simulator (refer to Section VII for further discussion on this).

D. Control-based RL

In our experiments, we compare our RL trajectory planning method to a direct RL control method. The control method mirrors the common approach to self-driving using an RL policy to directly control the actuators of the vehicle as in [6]–[10]. In the control-based method, Algorithm 2 is not required. Instead, the RL policy directly outputs the control c to be directly applied to the car.

To emphasise the advantage of our trajectory method over control, we keep nearly everything about the experimental setup to be the same when comparing the two. Specifically, everything about the observation space and reward structure is kept constant between the two, with the exception of control getting a +50 reward for success, instead of +10. Additionally, all hyperparameter settings shown in Table III and all algorithm parameter settings shown in Table I are kept constant between the two.

V. RESULTS

The main advantage of the trajectory planning approach that we demonstrate here is safety and comfort. That is, we hypothesized that by planning trajectories instead of directly controlling the car, we would simplify the learning process and experience fewer failures; and by using a PID controller to control the car, we would create a smoother drive than end-to-end control RL methods. We see from the results following that our hypothesis was correct.

TABLE IV

EXPERIMENTAL SIMULATION ENVIRONMENT PARAMETER SETTINGS

Parameter	Constant (C) or Variable (V)	Value(s)
Ego initial lane	V	{1, 2, 3}
Sim. Lane	V	{1, 2, 3}
Sim. initial x	V	$[20, 60]$ m
Sim. heading	C	with traffic
Ego target speed	C	72 km/h
Sim. speed	C	36 km/h
Road length	C	120 m
Vehicle length	C	5 m
Number sim. cars	V	{0,1,2}
Lane width	C	6 m
Number of lanes	C	3
Road geometry	C	straight

TABLE V

FAILURE RATE IN 100 STOCHASTIC EPISODES

Num. Sim. Cars	Control	Plan (Ours)
0	0%	0%
1	15%	0%
2	32%	7%

A. Safety

To showcase the safety improvement of our system, we run 100 stochastic (refer to Table IV for what was randomized) episodes with 0, 1 and 2 obstacle cars. We report the failure percent, which is any episode terminating in driving off the road or colliding with another vehicle. The results are reported in Table V. From these experiments, we see a dramatic increase in safety in our plan-based method over the direct control-based method. Both the control and plan-based method were trained on 2M steps.

One of the advantages of planning ahead is that we could theoretically apply higher-level logic to avoid failure scenarios. One way we could do this is to force the planner to replan if we foresee a dangerous outcome of following the trajectory. This we could do by resampling from the stochastic policy. Another possible way to ensure safe driving was proposed in an imitation learning trajectory planner proposal [12]. Here, the authors propose a projection of the chosen trajectory onto free space such that the change by projection is minimized. We believe incorporating a safety module into our approach could further decrease the failure rate and we leave this open to future work.

B. Comfort

By planning a trajectory and letting PID control the vehicle, we achieve lower average turning and acceleration than end-to-end control, as seen in Fig. 4. This translates into a smoother, more comfortable ride. These results are from the same experiments that produced Table V.

C. Demonstrations

Fig. 5 shows our simulation environment and demonstrates the capabilities of the system to plan within dynamic environments. Fig. 6 reveals some of the failure cases still experienced from the RL planner.

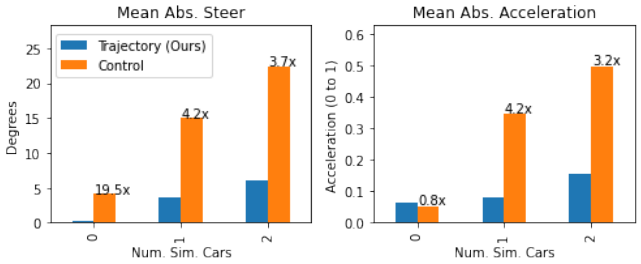


Fig. 4. Trajectory planning reduces unnecessary controls such as steering and acceleration. *Control* is the method which uses RL to directly control the vehicle. Note that some steer and acceleration is necessary for avoiding obstacles. The numbers above the *Control* bars represent by what factor *Control* exhibits more of the control than *Trajectory*.

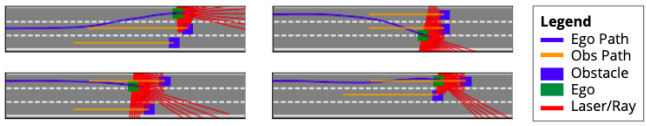


Fig. 5. Four dynamic environment instantiations demonstrating the agents ability to successfully overtake in traffic using our RL trajectory planner approach. Visualizations are shown at various progression of their episodes.

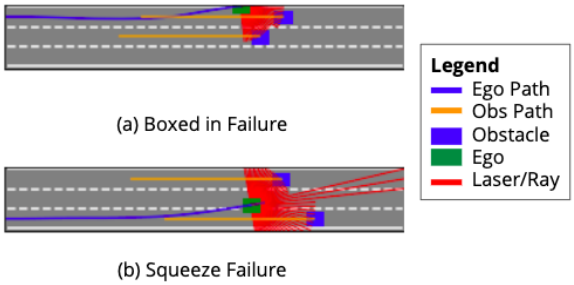


Fig. 6. Most common failure cases of our RL trajectory planner. Agent does not turn soon enough and so gets trapped (a). The agent wants to give the vehicles on both sides of it space and in the process, collides with one of them (b). Visualizations are shown near the end of their episodes (before failure).

Visit our website[1] for moving demonstrations.

VI. CONCLUSION

In this work, we propose a planning structure that combines the reinforcement learning-based trajectory planning and a PID controller for self-driving vehicles in dynamic environments. Instead of controlling the car using deep reinforcement learning directly, a higher level trajectory is generated by RL and then optimized through a lower level PID controller. We test the proposed approach in various scenarios and achieve a safer and more comfortable driving profile in the simulation.

VII. FUTURE WORK

In future work, we would like to generalize our approach to allow the agent to plan under varying settings of all

[1] https://sites.google.com/coad.net/riss/home

parameters outlined in Table IV. We believe that incorporating an LSTM into our approach would allow capturing the partially observable (POMDP) nature of driving and would generalize our approach. We also wish to extend our approach to incorporate a perception module directly into our policy via a CNN, as in the hybrid approach shown in Fig. 1(c), which would better take full advantage of the high capacity of our machine learning approach. We also believe that implementing the *safe PID* from [12] would further improve the safety of our system. For a more robust comparison of our work, we wish in future work to directly compare our results to imitation learning-based and non-deep learning (traditional) trajectory planning. Finally, we wish to apply our algorithm to the CARLA driving simulator and report our performance on the CARLA baselines.

VIII. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. We would like to thank the organizers of the Carnegie Mellon Robotics Institute Summer Scholars (RISS) program for making this experience possible.

References

[1] S. M. Grigorescu, B. Trasnea, L. Marina, A. Vasilcoi, and T. T. Cocias, "Neurotrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles," *CoRR*, vol. abs/1906.10971, 2019. [Online]. Available: http://arxiv.org/abs/1906.10971

[2] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. W. Mutz, T. Oliveira-Santos, and A. F. de Souza, "Self-driving cars: A survey," *CoRR*, vol. abs/1901.04407, 2019. [Online]. Available: http://arxiv.org/abs/1901.04407

[3] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[4] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," *CoRR*, vol. abs/1904.08980, 2019. [Online]. Available: http://arxiv.org/abs/1904.08980

[5] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *CoRR*, vol. abs/1710.02410, 2017. [Online]. Available: http://arxiv.org/abs/1710.02410

[6] X. Liang, T. Wang, L. Yang, and E. P. Xing, "CIRL: controllable imitative reinforcement learning for vision-based self-driving," *CoRR*, vol. abs/1807.03776, 2018. [Online]. Available: http://arxiv.org/abs/1807.03776

[7] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," *CoRR*, vol. abs/1807.02371, 2018. [Online]. Available: http://arxiv.org/abs/1807.02371

[8] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," 2016.

[9] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," 2019.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: an open urban driving simulator," *CoRR*, vol. abs/1711.03938, 2017. [Online]. Available: http://arxiv.org/abs/1711.03938

[11] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to drive using inverse reinforcement learning and deep q-networks," *CoRR*, vol. abs/1612.03653, 2016. [Online]. Available: http://arxiv.org/abs/1612.03653

[12] J. Chen, B. Yuan, and M. Tomizuka, "Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety," *CoRR*, vol. abs/1903.00640, 2019. [Online]. Available: http://arxiv.org/abs/1903.00640

[13] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," *CoRR*, vol. abs/1810.06544, 2018. [Online]. Available: http://arxiv.org/abs/1810.06544

[14] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," 2019.

[15] M. Bansal, A. Krizhevsky, and A. S. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *CoRR*, vol. abs/1812.03079, 2018. [Online]. Available: http://arxiv.org/abs/1812.03079

[16] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, "Agent57: Outperforming the atari human benchmark," 2020.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[18] W.-H. C. L. D. Christos Katrakazasa, Mohammed Quddusa, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research*, 2015.

[19] K. Macek, M. Becker, and R. Siegwart, "Motion planning for car-like vehicles in dynamic urban scenarios," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 4375–4380.

[20] T. Gu, J. Snider, J. M. Dolan, and J. Lee, "Focused trajectory planning for autonomous on-road driving," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 547–552.

[21] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *CoRR*, vol. abs/1604.07446, 2016. [Online]. Available: http://arxiv.org/abs/1604.07446

[22] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," 2019.

[23] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," 2019.

[24] A. Tampuu, M. Semikin, N. Muhammad, D. Fishman, and T. Matiisen, "A survey of end-to-end driving: Architectures and training methods," 2020.

[25] M. Wulfmeier, D. Z. Wang, and I. Posner, "Watch this: Scalable cost-function learning for path planning in urban environments," *CoRR*, vol. abs/1607.02329, 2016. [Online]. Available: http://arxiv.org/abs/1607.02329

[26] Y. W. K. Z. L Yu, X Shao, "Intelligent land-vehicle model transfer trajectory planning method based on deep reinforcement learning," *Sensors*, 2018.

[27] Á. Fehér, S. Aradi, F. Hegedüs, T. Bécsi, and P. Gáspár, "Hybrid ddpg approach for vehicle motion planning," in *ICINCO*, 2019.

[28] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *CoRR*, vol. abs/1907.10902, 2019. [Online]. Available: http://arxiv.org/abs/1907.10902

[29] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

Traffic Sign Detection and Localization on the Edge for HD Map Updating

Hunter Damron[1] and Christoph Mertz[2]

***Abstract*— In this paper, we present methods for traffic sign detection and localization which will be used to determine when traffic signs have been added or removed since the creation of a known high definition (HD) map. This work is part of a larger project which will use cameras on public buses to continuously monitor for HD map changes and update the map to reflect those changes. Our system uses a MobileNetV1 Single Shot Multibox Detector (SSD) to detect traffic signs and is designed to run in real time on the edge. We also use Colmap Structure from Motion (SfM) to localize the camera in a known HD map. Our object detector achieves 96.2% precision per real-world occurrence with 96.2% recall for sign detection in the Argoverse dataset. Together with the localization component, the detector can be used to identify video frames which contain map changes. When implemented on a public bus, this system will provide temporal guarantees of the traffic signs labelled in an HD map, and improve the safety of autonomous driving.**

***Index Terms*— Intelligent Transportation Systems, Localization and Mapping, Computer Vision**

I. INTRODUCTION

Self-driving cars have shown promise in improving the safety and efficiency of transportation [1]. In order to safely navigate, autonomous cars must obey posted signage and lane markings while also avoiding obstacles such as pedestrians and other cars. Dynamic obstacles have to be detected at runtime, but road signs and markings can be included in high definition (HD) maps, such as Argoverse [2], which provide precise information about the drivable area.

HD maps can improve the safety of self-driving cars by providing additional context for navigation. However, these maps are expensive and time consuming to construct because a sensor-laden vehicle must drive every street in the map. Additionally, because the highway infrastructure can change, these maps must be updated regularly to reflect changes.

Between map updates, outdated road artifacts can exist in the map for days to years, depending on the frequency of remapping. To solve this, the Carnegie Mellon NavLab group is developing a method to detect HD map changes using cameras onboard public buses [3]. Because these vehicles patrol the streets regularly, changes can be detected within days or even hours.

We aim for near real-time processing of the image streams provided by the bus, but we cannot sacrifice accuracy for speed. Hence, we use a two-step system which separates the analysis between the bus's computation-limited onboard computer and a nearby cloudlet server [4] which provides powerful computation resources. The onboard computer performs a quick comparison between the collected images and the existing map to determine if the map has changed. Images which contain map changes can be uploaded to the cloudlet for a more precise remapping or prioritized in future remapping efforts.

In this paper, we propose the detection and localization methods for one such onboard data filter which would locate new or removed traffic signs in the current map. We detect signs using a Single Shot Multibox Detector (SSD) [5] with a MobileNetV1 [6] backbone implemented by the TensorFlow [7] Object Detection API [8] and trained on the Mapillary Traffic Sign Dataset [9]. We then align images to the existing map using Colmap [10] to match detected signs with known signs in the map.

Because this stage acts as a preliminary filter, we require high recall to ensure the connected cloudlet is provided images of all possible map changes. False positive detections are of less concern because they will likely be removed by the second stage of the pipeline. However, we are restricted by the bandwidth of the connection between the bus, and false positives increase burden on the network. Therefore, we must discard enough frames to avoid overloading the network while also retaining enough information for the cloudlet to determine the location of all sign changes.

Instead of the recall in each individual frame, we leverage the temporal stability of the video streams from the bus and consider recall per real-world occurrence. The SSD is known to be weaker at detecting small objects [5] and would have difficulty detecting far away signs. However, as the bus approaches these signs they get larger in the image and, hence, easier to detect. Additionally, because buses make several loops of their route each day, we can combine multiple iterations for improved accuracy. This is especially important for differentiating if a sign has been removed or is just occluded.

The rest of this paper continues with a discussion of related work. Section III provides a detailed overview of the proposed approach. Experimental results of the system and each of its components are presented in Section IV. The paper concludes with a discussion of future work and the implications of the project.

II. RELATED WORK

Traffic sign change detection relies on two distinct components: localization and sign detection. Many solutions have been proposed for each.

[1]Hunter Damron is with the Computer Science and Engineering Department, University of South Carolina, Columbia, SC, USA. hdamron@email.sc.edu

[2]Christoph Mertz is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA cmertz@andrew.cmu.edu

For localization, Colmap [10] provides state of the art structure from motion and allows registration of new images in an existing map while ORB-SLAM2 [11] allows real-time simultaneous localization and mapping (SLAM). Other techniques incorporate data from other sensors such as LIDAR [12] and sonar [13]. Recently, deep learning has been used to augment SLAM techniques by extracting more robust features [14] and estimating scale for monocular SLAM [15]. In this paper, we use Colmap because we require features which can handle changes in scale and illumination.

Deep learning techniques have also proven successful in the object detection necessary for identifying traffic signs [16], people [17], and cars [18]. Transfer learning frameworks such as Detectron2 [19] make it easy to perform object detection by fine-tuning common models like Faster R-CNN [20] which are trained on standard datasets such as COCO [21] or ImageNet [22].

To achieve real-time object detection on embedded computers, simpler neural networks like YOLO (You Only Look Once) [23], [24] and the SSD (Single Shot MultiBox Detector) [5] are more efficient because they combine the region proposal and classification stages. Performance on mobile processors is further improved by using a MobileNetV1 [6] feature extractor as the backbone. However, these approaches sacrifice accuracy for efficiency. Arcos-García et al. provide an extensive comparison of the accuracy and efficiency of the state of art object detectors in [25]. Here, we use a MobileNetV1 SSD because it provides a suitable tradeoff between efficiency and accuracy. We also benefit from the availability of models pre-trained on the COCO dataset because we can fine tune an existing model rather than starting from scratch.

Automatic road change detection was first performed using satellite imagery [26]. However, modern methods use more precise sensors, typically vehicle mounted, to obtain the detail needed for autonomous driving. In [27], Pannen et al. use a particle filter to continuously monitor changes in road geometry using anonymized floating car data (FCD) provided by the car manufacturer. In [28], smartphones are mounted in service vehicles to continuously monitor for road distress.

III. SYSTEM OVERVIEW

A. Preprocessing of Argoverse HD Map

The Argoverse dataset [2] provides lane-level geometry and other information about the drivable area in Pittsburgh, PA and Miami, FL. Argoverse also provides data from nine cameras and a 3D LIDAR sensor mounted on the roof of an autonomous vehicle as it drives. Figure 2 shows a sample image included in the Argoverse dataset.

Because the bus route is known ahead of time, we first reduce the Argoverse dataset to a subset containing only the route. We then use Colmap [10] to create a sparse 3D model of that route using seven of the nine cameras, as in Figure 3. We also annotate each image with its GPS location and include SIFT features [29] so future images can be quickly localized in the map. Additionally, in order to reduce the size of the model, we downsample the 30 Hz image stream so that consecutive images are at least 5 m apart. For convenience, we align the model to the Argoverse coordinate system.

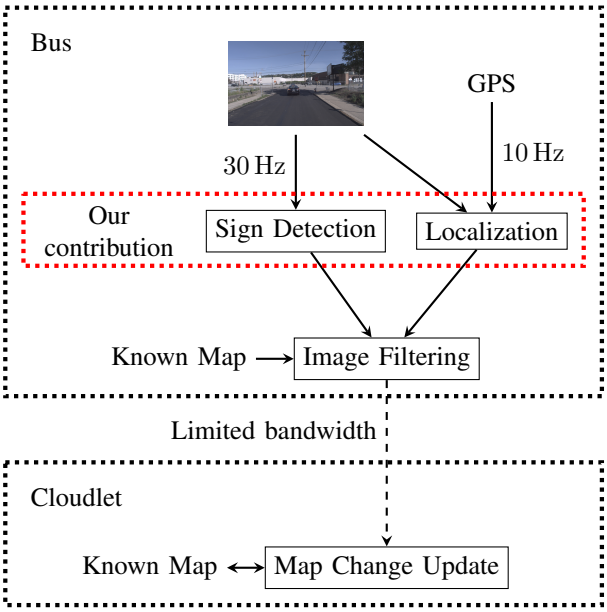


Fig. 1: System overview

Fig. 2: Sample Argoverse image in Pittsburgh

Next, we detect traffic signs in each image using Faster R-CNN [20] with Detectron2 [19] and add them to the map as in [30]. Although we could label these images manually, we follow [30] because their work will be the basis of the map update process running on the cloudlet server. The 3D map from Colmap along with the annotated signs will be used by the online filter to determine which signs are missing in the map and which signs have been removed.

B. Setup of Real-time Replay

In this paper, we simulate the conditions of the bus by performing real-time replay of a subset of the Argoverse data which was not used to build the map. To be consistent with the sensory input collected by the bus, we consider only one

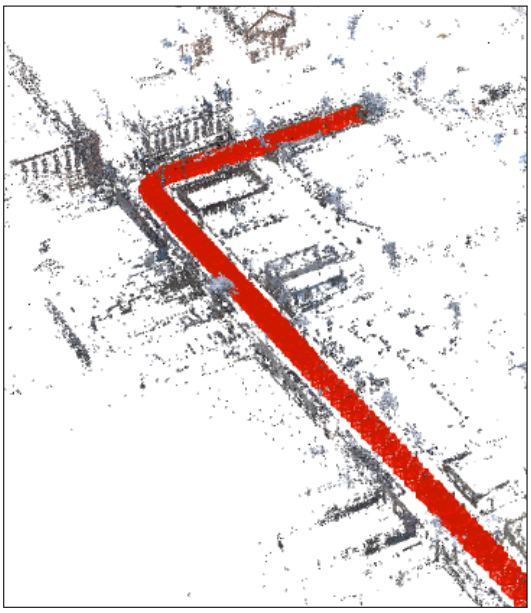


Fig. 3: Sparse reconstruction of a subset of the Argoverse dataset. Camera positions are shown in red.

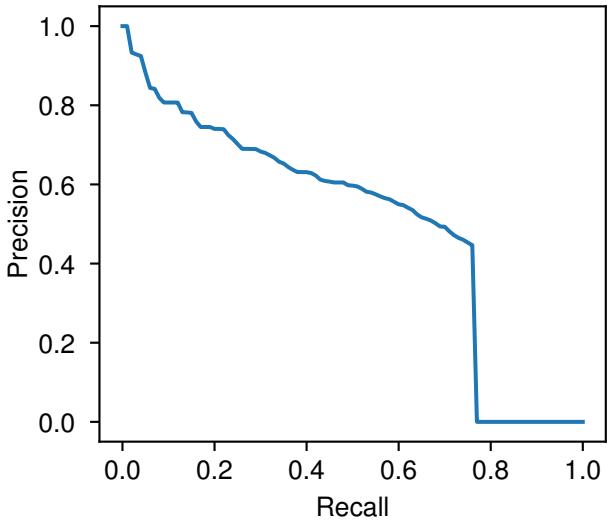


Fig. 4: Precision on Mapillary validation set objects larger than 225 sq px - AP 50.3%

of the nine cameras and reduce the resolution to 1920×1080. To simulate GPS, we add normally distributed noise to the Argoverse positions with average absolute deviation of $4.9\,\mathrm{m}$ [31]. Images are played back at $30\,\mathrm{Hz}$ and GPS at $10\,\mathrm{Hz}$. For compatibility with the bus computer, we use ROS [32] to manage the communication between the sensors and the detection node. As a result, our real-time replay is simply a ROS bag file.

C. Localization in HD Map

The first step in determining if any signs have been added or removed since the creation of the map is to localize the bus in the map. We do this by extracting features from incoming images and registering those features into the map using Colmap [2]. This would normally require an exhaustive comparison to every image in the map, but using the bus's GPS location as a hint we instead only consider several of the nearest images. Experimentally, 10 candidate images are usually sufficient, but we allow up to 30 candidates to be checked before moving to the next image.

D. Sign Detection

For real-time sign detection, we use a MobileNetV1 [6] SSD [5] implemented by the TensorFlow [7] Object Detection API [8]. In order to speed up training, we start with a model which has already been trained on the Microsoft COCO dataset [21]. Then, to detect traffic signs, we fine tune the model by training on the Mapillary Traffic Sign Dataset [9] for 100000 steps.

IV. EXPERIMENTAL RESULTS

A. Evaluation Criteria

In order to verify the project's success, we present specific constraints which must be satisfied by the system.

The limited bandwidth of the connection between the bus and the cloudlet provides a lower bound on the system's precision. Specifically, considering a typical 10 Mbps 4G network and 1920×1080 resolution images with JPEG compression to 10%, we can upload at most 2 images per second in the ideal case. However, the bus will be running up to 4 tasks in parallel and the connection may only be available say 80% of the time. As a result, the upload allowance is cut to roughly 0.4 images per second. In contrast, in the Argoverse dataset, 26 signs (excluding parking and street signs) are counted along a $544\,\mathrm{m}$ path, leading to an estimate of 0.7 required uploads per second. Hence it is impossible to upload images of every traffic sign in our setup. This discrepancy is the motivation for filtering sign changes on the bus.

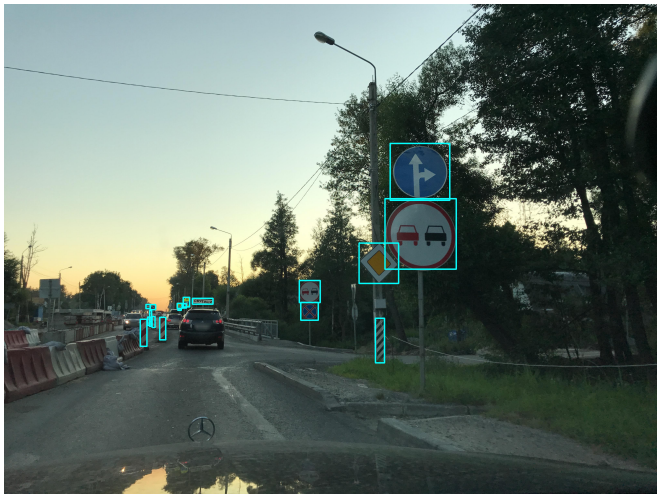
One way of reducing the number of uploads is to upload a few views of each sign rather than uploading every image which contains a sign. This also allows us to consider only large signs in our evaluation because far away signs will get closer in later frames. We use a threshold of 225 square pixels (in 300×300 image), which corresponds to a distance of roughly $8.5\,\mathrm{m}$ for a $75\,\mathrm{cm}$ stop sign in our setup.

Lastly, we want to ensure all relevant images are uploaded to the cloudlet. As an estimate of the allowed tolerance, we consider the 1.13 fatalities per 100 million vehicle miles travelled reported by National Highway Traffic Safety Administration in 2018 [33]. Assuming for the sake of an estimate that every death corresponds to a missed traffic sign, this means traffic signs are missed with probability on the order of 10^{-16}.

B. Sign Detection

We evaluate the sign detection system in two ways. First we consider the average precision (AP) of our detector on the Mapillary validation set with IoU $= 0.5$. For all signs larger than 225 square pixels, the detector yields 50.3% average

(a) Our detector



(b) Ground Truth

Fig. 5: Sample sign detection on MTSD validation set

precision, and Figure 4 shows the precision-recall curve. Figure 5 shows the detections on a sample image from the MTSD validation set.

Because we are primarily concerned with AP per real-world occurrence, we count the number of detections across frames in a small subset of the Argoverse dataset, considering an object detected if it is detected in at least one frame. Using this metric the system detects 46 of the 94 signs, yielding a recall of 90.2% with precision 48.9%. However, when we restrict the evaluation to take into account only signs larger than 225 square pixels (of which there are 26), this is improved to 96.2% recall with 96.2% precision.

As a result, after the bus route is cycled 12 times ($1-2$ days), the probability of a sign being missed is reduced to the order of 10^{-18}, lower than the estimated 10^{-16} threshold.

C. Localization

In its current state, localization takes roughly 13 s per image on a single CPU core, which is too slow for near real-time processing. To demonstrate the current capabilities,

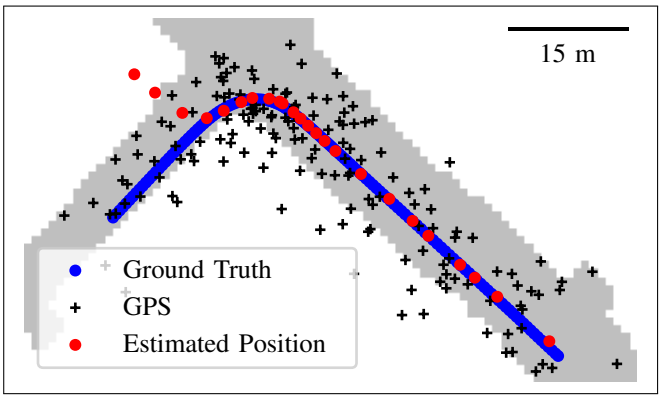


Fig. 6: Colmap localization on Argoverse dataset played at 5% speed

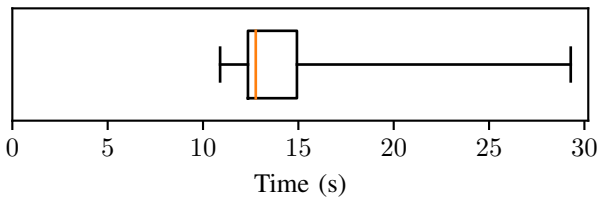


Fig. 7: Distribution of registration times on a single CPU core for a sample Argoverse segment with 74 images. The median time needed for image registration is 12.8 s.

we reduce the playback speed to 5%. Figure 6 shows the calculated trajectory compared with ground truth provided by Argoverse. The distribution of registration times are shown in Figure 7. Although the bus's onboard computer does not currently have GPU, we plan to add one in the future. With a GPU, the localization subsystem would benefit from a roughly 10× speedup and be usable in near real-time.

V. FUTURE WORKS AND CONCLUSIONS

In this paper we present a sign detection filter which fits into the larger "Bus on Edge" project of the NavLab group. The obvious next step is to implement this filter on the bus computer and integrate it with the more powerful sign change detection node running on the cloudlet server.

In order to improve the localization presented in this paper, we hope to streamline the system in C++ and avoid reloading the current map unnecessarily. We also plan to explore alternative localization techniques and offload some computation onto the cloudlet computer. For the detection subsystem, we plan to increase performance by tracking signs between frames.

Once our system is running on the bus and working with the cloudlet server, we will be able to provide continuous sign change monitoring of the roads patrolled by buses with minimal maintenance cost. The system could also be extended to other service vehicles which are confined to a specific route, such as garbage trucks and postal vehicles. By expanding the training dataset, other classes of map changes

could also be added to the detection process. Providing temporal guarantees on HD map features makes them much more reliable and useful for autonomous navigation.

VI. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. We would like to thank the organizers of the Carnegie Mellon Robotics Institute Summer Scholars (RISS) program for making this experience possible.

References

[1] P. Penmetsa, E. K. Adanu, D. Wood, T. Wang, and S. L. Jones, "Perceptions and expectations of autonomous vehicles – a snapshot of vulnerable road user opinion," *Technological Forecasting and Social Change*, vol. 143, pp. 9–13, 2019.

[2] M.-F. Chang, J. W. Lambert, *et al.*, "Argoverse: 3D tracking and forecasting with rich maps," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] C. Mertz, "Bus on the edge: Continuous monitoring of traffic and infrastructure," https://www.cmu.edu/metro21/news-and-events/metro21-events/2020/landslide-event.html, Apr. 2020.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[5] W. Liu, D. Anguelov, *et al.*, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, vol. 9905, 2016, pp. 21–37.

[6] A. G. Howard, M. Zhu, *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017.

[7] M. Abadi, A. Agarwal, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[8] J. Huang, V. Rathod, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, G. Neuhold, and Y. Kuang, "The mapillary traffic sign dataset for detection and classification on a global scale," 2019.

[10] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments," in *Robotics: Science and Systems (RSS)*, vol. 3, June 2007.

[13] S. Rahman, A. Q. Li, and I. Rekleitis, "Sonar visual inertial SLAM of underwater structures," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5190–5196.

[14] R. Kang, J. Shi, X. Li, Y. Liu, and X. Liu, "DF-SLAM: A deep-learning enhanced visual SLAM system based on deep local features," 2019.

[15] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular slam with learned depth prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6565–6574.

[16] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2110–2118.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[18] C. Mertz, S. Qian, and J. Chiang, "Improving rush hour traffic flow by computer-vision-based parking detection and regulations," 2020.

[19] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015.

[21] T.-Y. Lin, M. Maire, *et al.*, "Microsoft COCO: Common objects in context," 2014.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[24] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020.

[25] A. Arcos-García, J. A. Álvarez García, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, Nov. 2018.

[26] D. Klang, "Automatic detection of changes in road databases using satellite imagery," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (IAPRS)*, D. Fritsch, M. Englich, and M. Sester, Eds., vol. 32, no. 4, 1998, pp. 293–298.

[27] D. Pannen, M. Liebner, and W. Burgard, "HD map change detection with a boosted particle filter," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 2561–2567.

[28] C. Mertz, S. Varadharajan, S. Jose, K. Sharma, L. Wander, and J. Wang, "City-wide road distress monitoring with smartphones," in *Proceedings of ITS World Congress*, Sept. 2014.

[29] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[30] Y.-C. Kuo, S. Yu, Y.-T. Lin, C. Mertz, and J. Dolan, "Updating HD maps with videos from vehicles," https://mscvprojects.ri.cmu.edu/2020teamg/project/, May 2020.

[31] F. van Diggelen and P. Enge, "The world's first GPS MOOC and worldwide laboratory using smartphones," in *International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+)*, Sept. 2015, pp. 361–369.

[32] M. Quigley, K. Conley, *et al.*, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009, software available from www.ros.org.

[33] *2018 Fatal Motor Vehicle Crashes: Overview*, National Highway Traffic Safety Administration, Oct. 2019.

Three-stage 3D Shape Completion for Autonomous Vehicles

Sombit Dey[1], Chen Fu[2] and John Dolan[3]

***Abstract*— Scene understanding, which includes shape completion, is one of the major perception tasks for autonomous vehicles. 3D LIDAR algorithms provide crucial information about the surroundings in changing lighting and weather conditions. Having a complete shape of the object allows us to obtain high-level knowledge of the surrounding, which can be used for planning in complex scenarios and improving detection algorithms. Previous research use a object detector to extract the object point, but our work adds to this attempt to improve the accuracy using semantic masks and depth completion. We propose a three-stage deep regression model for 3D object detection and shape completion using sensor fusion from monocular images and sparse LIDAR data. Previous research on 3D object detection and completion is performed directly on sparse LIDAR data. The core novelty of our approach is the usage of depth completion and image features in our shape completion approach on KITTI dataset using weights trained on a synthetic dataset. We also demonstrate the ability of the method to generate realistic shapes of vehicles.**

***Index Terms*— RGB-D Perception, Deep Learning for Visual Perception**

I. INTRODUCTION

Scene understanding with dynamic obstacles for autonomous vehicles has been a long-time research problem. It is necessary for autonomous vehicles to improve their perception algorithms for real-life application and to deal with challenging and unpredictable situations. Autonomous vehicles need to sense both visual and geometric aspects of the environment to infer the semantic information required for driving. To make informed decisions (such as planning and behaviour prediction), it is of utmost importance to establish as thorough scene understanding as possible. It is imperative to have an accurate pose and heading angle about the other vehicles for an ego vehicle, which is crucial for path planning tasks.

Shape completion of obstacles can come into use for a high-level understanding of the scene. Shape completion helps the ego-vehicle to have a better understanding of the behaviour of the other cars in the scene. However, shape completion is a very challenging task for several reasons. Firstly, the LIDAR(Light Detection and Ranging) scans are highly sparse and irregular. Moreover, it is a non-trivial task to improve the accuracy of detection and shape completion without ground truth annotation of the 3D shape. In this paper, we focus on the specific problem of shape completion of objects using monocular images and sparse LIDAR data for autonomous vehicles.

[1]Sombit Dey is a 2020 Robotics Institute Summer Scholar and a student at Indian Institute of Technology, Kharagpur, India sombitdey888@@iitkgp.ac.in

[2]Chen Fu is a PhD student at Carnegie Mellon University cfu1@andrew.cmu.edu

[3]John Dolan is a professor at Robotics Institute, Carnegie Mellon University jdolan@andrew.cmu.edu

A. Related Work

1) Shape Completion: 3D shapes are complicated entities, unlike images, which can be expressed using a fixed matrix. 3D shape perception is long-standing and fundamental in computer vision. A large amount of work is focused on 3D reconstruction using multi-view images, which is an ill-posed problem as many different input data with different shape and texture generate the same image. Shape completion has been tried as an optimization problem using energy functions for shape estimation [1], but lacks to ability to generate a variety of shapes. But with the recent development in deep-learning, 3D shape completion can be formulated as a data-driven or learning-driven problem. The former usually rely on learned shape priors and formulate shape completion as an optimization problem over the corresponding input. Learning-based approaches [8], on the other hand, often learn the shape in a supervised manner from synthetic data. Shape completion [3,9,10] has been used recently for predicting the shape of a 3D object from 2.5D LIDAR scans. All previous research involves shape completion [1]–[3] on sparse LIDAR data. The results obtained using sparse data fail to generalize the shapes and have lower precision in case of occlusion. D. Stutz et al. [3] used variational auto-encoder network for shape completion. They introduced the notion of using two networks for shape completion, with weights learned in a synthetic dataset, which is then transferred to real-world scenarios.

2) Semantic Segmentation: Image segmentation is essential for visual understanding with various applications. Semantics segmentation has been carried out using deep learning models, assigning a class to each pixel. Many related works [4], [5] focus on the segmentation of monocular images using CNNs for image segmentation. O. Ronneberger et al. [6] have proposed a Unet architecture for image segmentation.

3) Sparse LIDAR Depth Completion: Depth completion aims to recover dense depth maps from sparse depth measurements. Depth Estimation from monocular images has been proposed using Deep CNN's [7], [8]. These methods suffer from over-fitting and do not generalize well in other scenes than the training data. LIDAR-based supervision improves depth image consistency [9]. Depth completion from a single RGB and sparse LIDAR scan has been proposed using a self-supervised method [10]. Features from both image and LIDAR are concatenated to achieve feature aggregation; thus

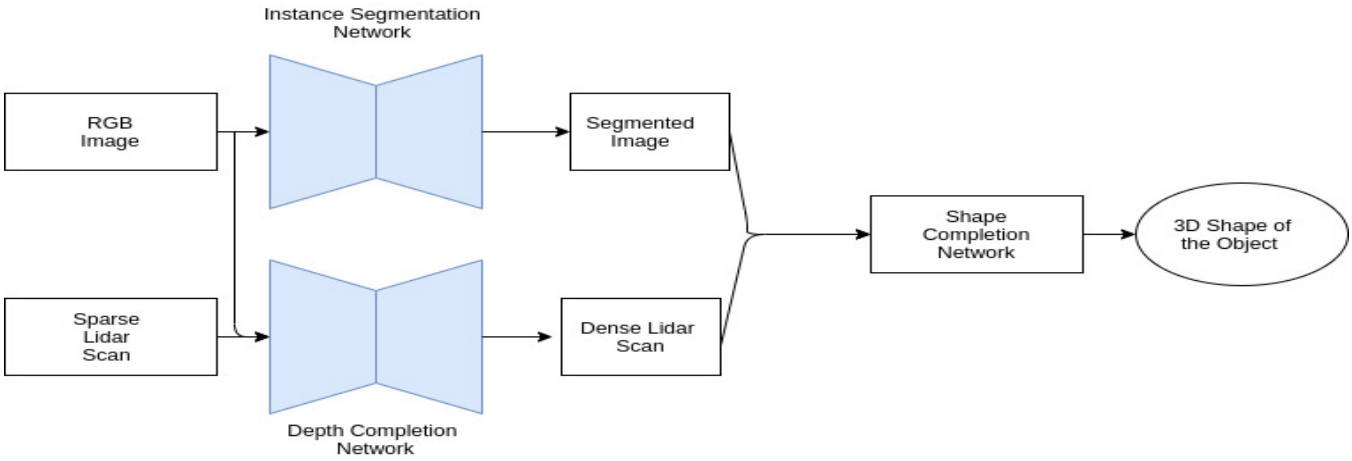


Fig. 1: Pipeline Description.

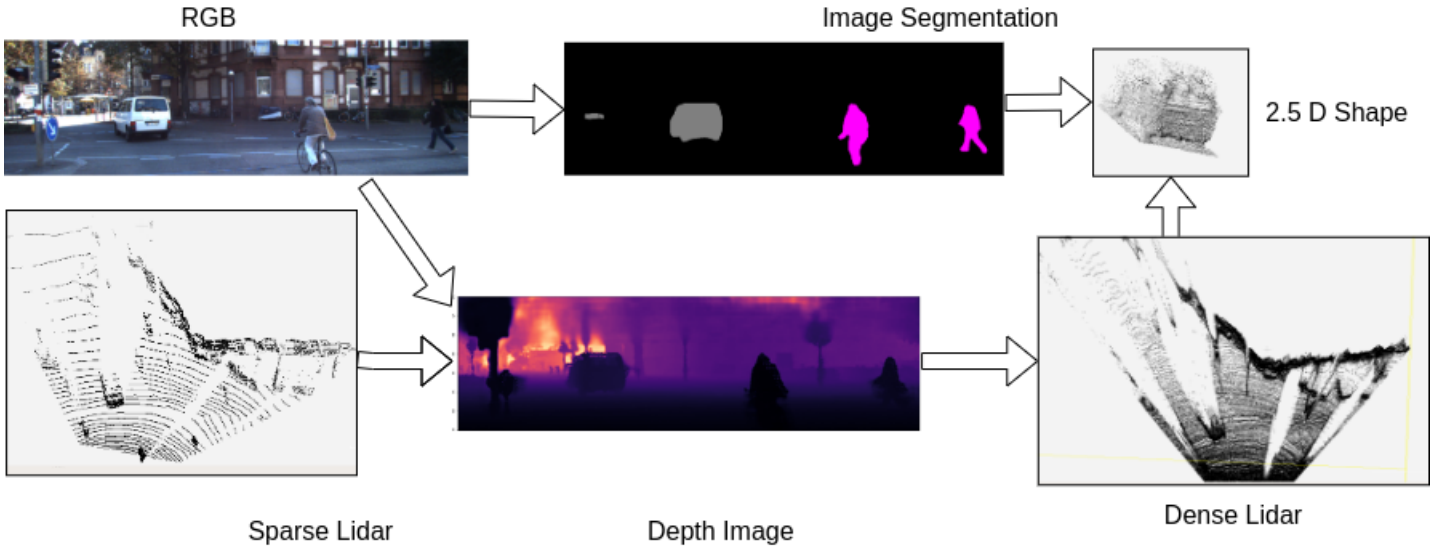


Fig. 2: Dense 2.5D LIDAR Data from RGB and Sparse LIDAR

the network is able to interpolate the data between the sparse depth maps from LIDAR scans.

B. Our Contribution

LIDAR can sense the scene geometry and depth accurately, whereas RGB images gather the texture and semantic information. Therefore, we propose a novel pipeline for object detection using shape completion with both RGB images and LIDAR scan. Our 3D shape completion model leverages pixel-level segmentation of RGB images and depth completion of the LIDAR scans. Our three-step network firstly uses instance segmentation on the RGB for mask generation of the objects using an encoder-decoder network. Secondly, the network also provides depth estimation using sensor fusion of the RGB and LIDAR data, allowing us to generate a semi-dense point cloud representation of the surroundings. Lastly, a shape completion model is used, converting the semi-dense point cloud to a complete 3D model, by learning a latent representation of the objects. Using a more dense point cloud along with additional RGB information makes the prediction more robust to noisy data and achieves more accurate depiction of the object. We apply a semantic mask from image data instead of the traditional bounding box, to obtain the partial shape of the object, thus reducing noise, as depicted in Fig.1 with relevant modules and outputs. We also propose a transfer-learning methodology for shape prediction of the depth-completed objects' shape using shapes of CAD models, which allows us to transfer knowledge to real-life scenarios. In brief, our contributions are 1) Leveraging image information for densification of LIDAR data, further used for shape completion; 2) Fusing sensor data from image and LIDAR data for shape completion of objects; 3) Utilizing semantic information rather than a bounding box for 3D shape prediction.

II. Overview of the Method

We propose a three-stage object shape completion, illustrated in Fig.1 and Fig.2. A variational autoencoder network is used to learn the synthetic shapes. This shape-prior network is able to predict shape using occupancy grid

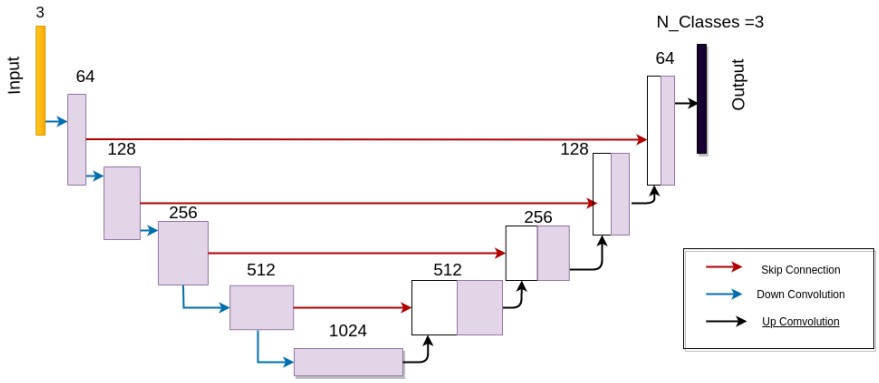


Fig. 3: Unet Architecture

Fig. 4: Output of Unet, for mask generation

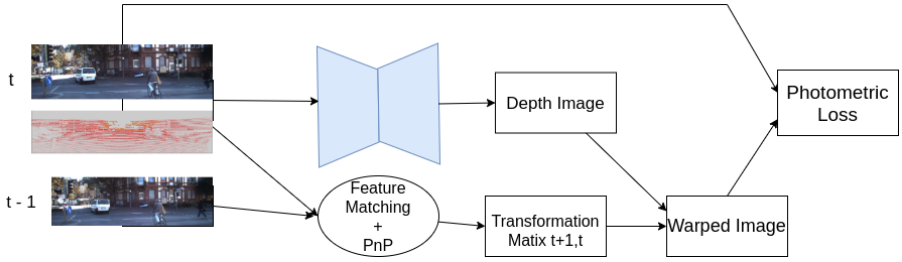


Fig. 5: Overview of Depth Completion pipeline

Fig. 6: Output of Depth Completion

representation, learning a latent space representation which can be used to generate new outputs. A shape-inference model is used to transfer these weights from synthetic data to the KITTI dataset. A densified point cloud representation of shapes is obtained using depth completion. The input RGB+LIDAR data is used to predict a depth map which is then re-projected into the world frame to obtain a dense pointcloud. The loss function for depth completion is modelled using the equation below a) L1 Loss : Calculated on data points where ground truth depth from the LIDAR depth map is available. b) Photo-metric Loss: The relative rotation and translation vectors are estimated between two nearby images using the predicted depth maps using SURF feature matching. c) Smoothness Loss : A Sobel edge filter is used to calculate the edges in the depth map. For ensuring a smooth depth map, a loss is associated with edges in the depth map.

$$L = \alpha_1 L_{L1} + \alpha_2 L_{photometric} + \alpha_3 L_{smoothness}$$

We apply an encoder-decoder network for pixel-level segmentation of the image using a Unet Architecture. The main advantage of Unet architecture is the shorter inference time and its ease of modification according to the need. The image is used to generate a high-dimensional feature vector, with features aggregated at multiple levels. The decoder takes a high dimensional vector to generate a semantic segmentation mask. Fig.3 shows the network architecture with feature size. Semantic masks from RGB images are used to segment out objects in the scene, reducing the noisy data points when compared to 3D bounding box segmentation of objects, shown in Fig.4 with mask in pink. Reduction of noise in the input data makes the observation more structured, resulting in better convergence to a latent space. A good convergence to a latent space is very crucial for the decoder, to ensure the generated models are sensible and follow a given structure. We train a new encoder to inlay the observation into the latent space from using a new network, called the shape-inference network. The shape-inference network is trained to learn the new embedding from the densified LIDAR dataset to the latent space. The encoder, $z(x, w)$, is updated to learn weights w from observation set x to generate the latent variable z. The decoder network is represented by $p(y|z)$, where y is the predicted shape with latent variable z. The loss function while training shape inference is

$$L = -\sum_{x} \ln(p(y = x|z)) - \ln(p(z))$$

where x is the voxels which are observed in the observation sample. The prior $p(z)$ follows a gaussian distribution, and the negative log of the probability function is representative of the variance of the distribution. This term is responsible for the regression for the latent space distribution.

A. Data

The Unet encoder-decoder architecture is trained on 8000 images on the KITTI-Tracking Dataset with 90:5:5 ratio of split data for training, validation and testing. The depth completion network is trained on the KITTI-Odometry dataset on 20,000 images without using the ground truth annotations of KITTI dataset [11]. The LIDAR data are filtered, keeping the points in the field of view of the camera, and rejecting the points that are far away and have low reflective values. The dense pointcloud data are obtained using the mask from

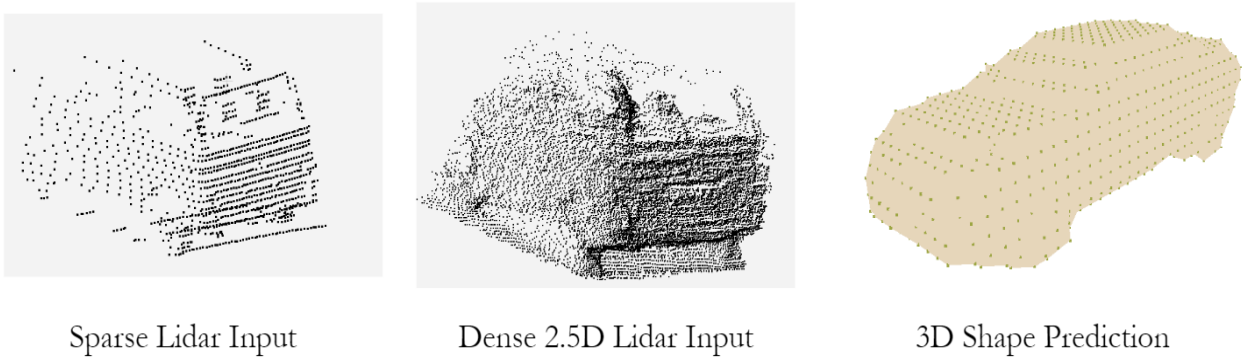


Fig. 7: Shape Completion Results

the instance segmentation model (Fig 4). A nearest neighbor clustering method is used to filter out the noise points due to erroneous depth values on the edges of the objects. The pointcloud is then normalized and voxelized into resolution 24 x 54 x 24. The angle is extracted from ground truth annotation, due to lack of a detection module. The generated occupancy grid is used as training dataset. The total number of 3000 cars are extracted for the shape completion model.

B. Training

1) Unet: The Unet Architecture is trained on the KITTI data set, with cross-entropy loss functions. The car weight in the loss function is higher than background resulting in under-segmentation during the calculation of cross-entropy loss function. The model is trained using batch-size 6 using SGD optimization. Data augmentation and learning rate decay is carried for better generalization of the model.

2) Depth Completion: The shape completion network by Fanchama et al [12] is trained on RGB-D input using batch size 4. The weights are initialized with the parameters of the ResNet-34 model. The network is trained using the self-supervised method, using images chosen randomly from 6 nearest images, excluding the current frame in the KITTI training sequence. During the training sequence, the input to the network is $RGB_t + D$ and RGB_{t-1} but during the inference time only $RGB_t + D$ is required.

3) Shape Completion: As proposed by D. Stutz et al., we use a shape prior in our network architecture for learning shape completion. We train the shape inference model on the densified LIDAR dataset (Fig. 4). The decoder part of the network is retained from the shape-prior pre-trained model and is frozen during the training process. An additional space representation is provided to the network for better convergence. Due to the lack of ground truth shape, using a signed distance field is difficult. A 3D space representation of the points is provided to the encoder. The loss is calculated on the points observed in the predicted shape. A latent space regularization term is added to the loss equation to restrict the decoder to predict shapes alike to the shape-prior.

III. Results

The pixel level segmentation using the Unet architecture from the above methodology is tested against Mask-RCNN, in the table below.

	Time (ms)	Accuracy
Unet architecture	68	85.3
Mask-RCNN	220	90.

Mask-RCNN has better accuracy in predicting pixel-level segmentation for cars than Unet, inferred from the table above. But the Unet architecture has smaller inference time and can be easily modified for the training without bounding box annotations. The depth completion results are shown in Fig.7. The depth images lack consistency on the edges or during abrupt changes in the depth. Therefore the resultant point cloud often is noisy, which is rectified by a nearest-neighbour clustering algorithm. A careful selection of hyper-parameters is necessary to avoid high-level image features in the depth-image.

	RMSE
Dense Dataset	0.63
Sparse Dataset	0.69

It is very challenging to compare shape completion due to lack to ground truth annotations. The RMSE values are calculated using the fact that the points in the observation should be on the surface of the predicted 3D shape. The error function is calculated using distance of the points to the surface of the predicted shape. Our approach has a better RMSE values compared to the earlier approach using sparse LIDAR scan even though the number of points in our observations is much higher. Hence we can conclude that our approach can generalize better when compared to the network when trained on a sparse LIDAR scan dataset.

IV. Conclusion and Future Works

In this work, we propose a methodology using both LIDAR and image features to generate a 2.5D point cloud representation of objects. A variational autoencoder network is used to learn the new weights to generate 3D shapes of objects using the generating model trained on a synthetic

dataset.

As part of future work, the integration of a 3D shape detector can be considered to make the pipeline not dependent on ground truth annotations. Along with object detection, insertion of the completed shape of the vehicle into the point cloud would be another important aspect of this research to allow these detected shape to be used for path planning.

V. ACKNOWLEDGMENT

This work was supported by the CMU Robotics Institute Summer Scholars (RISS) Program. The authors would also like to acknowledge the support of the RISS organizers: Dr. John M. Dolan and Ms. Rachel Burcin

REFERENCES

[1] F. Engelmann, J. Stückler, and B. Leibe, "Joint object pose estimation and shape reconstruction in urban street scenes using 3d shape priors," vol. 9796, 09 2016, pp. 219–230.

[2] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] D. Stutz and A. Geiger, "Learning 3d shape completion under weak supervision," *CoRR*, vol. abs/1805.07290, 2018. [Online]. Available: http://arxiv.org/abs/1805.07290

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017, cite arxiv:1703.06870Comment: open source; appendix on more results. [Online]. Available: http://arxiv.org/abs/1703.06870

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[7] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth prediction," October 2019.

[8] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv e-prints*, vol. abs/1812.11941, 2018. [Online]. Available: https://arxiv.org/abs/1812.11941

[9] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse lidar data with depth-normal constraints," 2019.

[10] D. Stutz and A. Geiger, "Learning 3d shape completion under weak supervision," *CoRR*, vol. abs/1805.07290, 2018. [Online]. Available: http://arxiv.org/abs/1805.07290

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," 2019.

Extracting Vital Waveforms from Microvascular Videos

Emma Erickson[1], Robert Edman[2], Micheal Pinsky[3], Hernando Gomez[3], and Artur Dubrawski[2]

***Abstract*— We introduce a machine learning method for uncovering underlying signals common to concurrently collected videos and vital signs. Sublingual microvascular videos are one method for monitoring patients undergoing surgery. These videos display red blood cell movement in the capillary bed beneath the tongue, analysis of which may indicate patient status and predict negative outcomes such as hemorrhage. Likewise, sensors both invasive and noninvasive can predict the onset of undesirable outcomes using collected metrics such as heart rate, blood pressure, and cardiac output. Because microvascular data is optical and therefore noninvasive, being able to predict invasive measurements or limit the number of necessary sensors by replacing them with a single video is desirable. We trained a set of random forest regressors using various featurizations of microvascular video data paired with a set of target vital waveforms. The models are trained and tested on a dataset of microvascular videos and vital waveforms collected from pigs exposed to hemorrhage in a laboratory setting. R^2 scores reveal motion features in the videos could explain 10-15% of variance in invasively measured blood pressure, and 15-20% of variance in continuous cardiac output and Sv02. Heart rate and cardiac output variance could be predicted at 57% and 41% respectively using frequency analysis of pixel intensities. In total our contributions are threefold, and include uncovering key meta information to synchronize the existing dataset, extracting information on invasive vital signs using only noninvasive videos, and identifying the predictability of particular vital signs through microvascular videos.**

***Index Terms*— Medical Systems, Micro-circulatory Analysis, Video Processing, Critical Care**

I. INTRODUCTION

More than 50 million surgeries are performed annually in the United States [1]. Some of these procedures, though common or even routine, will result in hemorrhaging, or unwanted bleeding. Hemorrhage during surgery is associated with worse outcomes: increased mortality, longer hospital stays and greater financial cost [2]. To preemptively predict these outcomes, sensors both invasive and noninvasive are used to track a patient's status. Collected waveforms such as those collected from arterial (ART) and central venous (CVP) catheters, as well as derived metrics such as cardiac output (CO), can indicate the onset of this undesirable outcome so that it may be quickly addressed [3].

[1]Emma Erickson is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA emmaee2@illinois.edu

[2] Robert Edman and Artur Dubrawski are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA redman,awd@andrew.cmu.edu

[3] Micheal Pinsky and Hernando Gomez are with the Department of Critical Care Medicine, University of Pittsburgh, Pittsburgh, PA, USA pinsky@pitt.edu , gomezh@upmc.edu

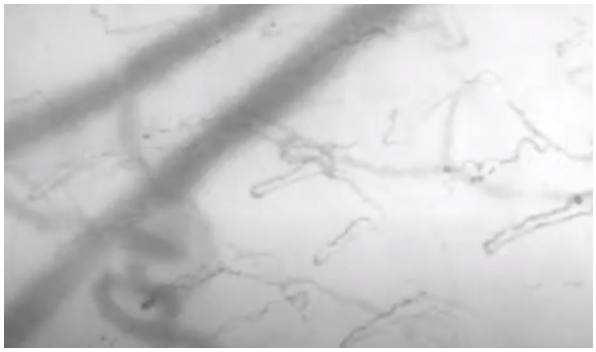


Fig. 1. A screenshot of a sublingual capillary bed from a microvascular video collected on an SDF device.

Microcirculatory monitoring is a technique for measuring the perfusion of small vessels, used to shed insight into the status of critically-ill patients. In normal, healthy patients, the capillaries display near constant perfusion [4]. Stopped or intermittent flow can indicate hemorrhaging, sepsis and other organ dysfunction, while homogeneous flow is linked to successful resuscitation and improved outcomes [5], [6]. Realtime bedside evaluation of microcirculatory flow is desirable to predict and respond to changes in patient status.

Sidestream dark field (SDF) imaging is one technique for monitoring microvascular blood flow. Using an RGB video camera, magnifying lens, and green LED ring for optimal red blood cell absorption, sidestream dark field imaging captures a network of perfused capillaries with varying levels of red blood cell flow. These videos are difficult to analyze due to noise, sensor drift, defocusing and the subsurface scattering of light. As a result, though real-time computer aided analysis of vessels is possible, it is limited by manual or semi-manual oversight [7]. In practice the gold standard remains manual evaluation. Existing and proposed techniques which have automated microvascular analysis focus on vessel extraction and classification (i.e. no flow, intermittent, continuous) without additional links to physiological states.

Optical instrumentation and subsequent video processing is a noninvasive way to collect medical data. From apps capable of detecting heart rate from cell phone video to red light sensors in health trackers monitoring blood oxygen levels, these trends indicate that common vital waveforms are embedded within noisy visual data. Metrics proven useful in revealing patient status, such as cardiac output and blood pressure, are commonly measured on other sensors alongside microvascular monitoring. In this paper we investigate directly connecting these two measurements. By featurizing both the blood flow videos (Fig. 1) and simultaneously

collected vital sign measurements, we implement random forest regressor models to understand the predictability of waveform data from video.

II. RELATED WORK

A. Signal Analysis

Machine learning has been leveraged in the medical field to make sense of the immense amount of data medical professionals are asked to interpret. Automated analysis of signals is of particular interest, as such processes save time, reduce manual errors, and eliminate subjectivity. Electroencephalogram (EEG) analysis is one area of research exemplifying this trend. For example, the problem of classifying an EEG waveform as normal or abnormal has been addressed in the past both by traditional machine learning tactics (RF, kNN) as well as with deep learning models [8], [9]. Both strategies outperform random guessing, though the standard remains analysis by trained clinicians.

Similar random forest methods have been utilized with success for hemorrhage prediction. In particular, invasive measurements reliant on catheters to measure blood pressure outperform noninvasive methods in how quickly they detect hemorrhage [3]. Being able to predict and respond to negative outcomes is critical to successful care, and vital signs offer this utility at bedside.

B. Video Analysis

It is understood that microvascular perfusion is altered in patients in critical condition [10]. Though not as mature as vital analysis in its bedside applications, microvascular analysis has revealed information in its own right. Studies in sepsis and shock show microcirculatory changes to be related to adverse outcomes [11]. To this end, the automation of microvascular analysis is a desirable goal.

Research on microcirculation is generally focused on vessel extraction and the quantification of perfusion. AVA Microtools has presented a real-time point of care tool for these analyses, but the gold standard still relies on manual aid [7]. Microvascular video data from hemorrhaging experiments has been analyzed with a focus on automating vessel extraction and flow type (absent, sluggish, continuous, etc.) [12]. But as a side product of the video stabilization process, this work also was able to recover some information on physiological measurements [13]. Namely, pixel intensities over time could be decomposed into frequencies representing the heart rate and respiratory rate of the subject. Figure 2 demonstrates results of repetition of these experiments, where a random forest regressor trained on microvascular videos is able to for the vast majority of samples predict heart rate within 10% accuracy.

Despite interest in and intuition that both microvascular video data and time series vital data contain relevant information to patient status, analysis has remained an either/or with connections between the two arising as side effects, without exploration into whether vitals beyond heart rate and respiratory rate are attainable visually.

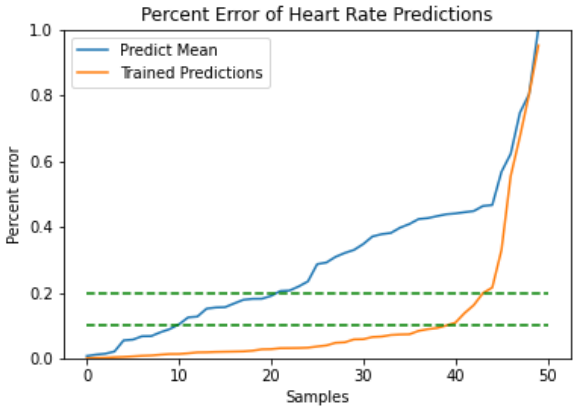


Fig. 2. Percent error in extraction of HR from pixel intensity frequencies of microvascular videos. The values are the percent errors sorted lowest to highest. The blue line shows error when the mean value is predicted each time, the orange are predictions made by the random forest regressor.

III. METHODS

A. Description of Data

To examine the recovery of a latent variable common to microvascular videos and vitals recorded in parallel, we used a dataset collected through a critical care experiment on hemorrhaging. Healthy pigs were anesthetized and then kept stable for a 30 minute baseline period. This is followed by induced bleeding at a rate of 20 ml/min for approximately 2 hours, and then by resuscitation. Sublingual microvascular videos are captured using the SDF device several times across these stages. In total, the video dataset consists of 52 480 x 720 pixel videos, approximately 20 seconds each at 30 fps.

In addition to the videos, CSV files record time series data through the duration of the experiment. 'Waveform' files include directly measured vitals such as ECGs and arterial pressure, and are collected throughout the experiment at 250Hz. 'LiDCO' files contain metrics calculated beat-to-beat (approximately 1-2Hz) from the waveform data. Tested waveform metrics are the arterial catheter (ART), electrocardiogram (ECG), continuous cardiac output (CCO), central venous pressure (CVP), pulmonary artery pressure (PAP), plethysmographic (Pleth), mixed venous oxygen saturation (SvO2), and oxygen saturation (SpO2). Tested LiDCO metrics are mean arterial pressure (MAP), systolic pressure (Sys), diastolic pressure (Dia), pulse pressure (PP), cardiac output (CO), stroke volume (SV), heart rate (HR), heart rate variability (HRV), systolic pressure variation (SPV), stroke volume variation (SVV), pulse pressure variation (PPV) and dynamic arterial elastance (PPV/SVV).

B. Synchronization

Video and vital timestamps were aligned so each 20-30 second video was matched with the corresponding measured vitals. The dataset was unpaired prior to these experiments, and time synchronization was necessary between the two.

Videos included a timestamp based on the 24-hr clock. Waveform data included a time column relative to the beginning of the 30 minute baseline period. Alignment was based off of a stabilization offset documented in experiment annotation.

C. Data Preprocessing and Featurization

In order to connect the microvascular videos to the vital signs through a machine learning model, the data was first translated into a more readily usable format. Two strategies were used to featurize the videos, which were all at least of size (480, 720, 600).

First, a three-dimensional convolutional method was used to extract the motion features of the videos. This pipeline had previously been used to classify microvessels by their level of perfusion [12]. Accepting the (480, 720, 600) videos as input, it generates a (480, 720, 100) histogram map, describing the motion features as a 100 bin histogram per pixel. Examining this output, pixels with higher total values summed across the bins of the histogram corresponded to moving pixels rather than background. These histogram sums describing motion features were used to train and test the first round of experiments.

The second strategy of video featurization revolved around pixel intensity. Throughout the video, the average of both background and vessel pixels per frame oscillated over the total run-time. These oscillations could be decomposed into frequencies in ranges corresponding to reasonable respiratory rate [0.1, 0.5] Hz and heart rate [0.5, 5] Hz. Both the raw average of pixel intensity per frame and its Fourier transform were used as video features in experiments.

The waveform and LiDCO data also required considerations. Because of the short time frame of the videos and the limited changes occurring in that window, the full videos were mapped to a single vital data point. This was chosen to be the average of the vital over a 1 minute interval beginning 20 seconds prior to the start of the video. While this made some vitals, namely ECG which encodes important information in the period between spikes, useless, for the majority it was necessary to smooth out fluctuations in the data due to respiration, heart beat, or gaps in the data.

Two videos contained completely invalid vital sign data during the minute surrounding their capture. They were removed from the experiments because they did not have these vital truth values.

D. Experimental Setup

Three experiments were set up to test each waveform and LiDCO vital. In each a random forest regressor was implemnented via scikit-learn. The regressors were trained to predict a target vital sign's average value at the time of the video recording. Each random forest regressor contained 100 trees with no specified maximum depth.

In the first experiments - motion features - each histogram map corresponded to one video. The histograms were summed into the total of the motion features for each pixel, and then these pixel values served as the input for the regressor. The next experiments examined the average pixel intensities over time, so each frame was described by its average pixel intensity, and each video was described by that value per frame. The final set of experiments took the time series nature of the video into account by running the regressor on the FFT of the averaged pixel intensity per frame series composed in the previous experiment. The complete fast Fourier transform, as well as the highest energy frequency over the total frequency range, the respiratory rate range - [0.1, 0.5] Hz, and the heart rate range - [0.5, 5] Hz were all tested.

The models were trained and tested on a one category of video features with the vital signs serving as truth data. Each experiment utilized 5-fold cross validation. The data was tested on each fold after being trained on the remaining four. The resulting predictions were scored using R^2 to determine the percentage of variance in the dependent variable which the independent variable could explain. An $R^2 = 0$ indicates guessing a constant each time, and $R^2 = 1$ would be a perfect regressor. The R^2 score may be negative for worse than random performance. The R^2 score was averaged across folds to produce the final score for a given type of video feature on each vital sign.

IV. RESULTS

The contributions of this project are threefold. Past research using the critical care pig bleed dataset has involved either analysis of microvascular videos or the analysis of the time series vital signs on their own, so synchronization has improved the dataset for future work combining videos with vital signs. Next, exploration of the 20 vital signs collected in parallel with the videos makes a case for the visual predictability of specific signs. Finally, the extent of prediction of these key vitals using the specified methods and featurizations is quantified.

A. Motion Features

The outcome of the motion feature models are summarized in the second column of Table 1. Overall, a large majority of the variance was not accounted for by these motion features across target vitals. The vitals that did have an R^2 value greater than 0 (some percentage of variance accounted for) were ART, CVP, MAP, systolic pressure, diastolic pressure, CCO and SvO2. The majority of these vital signs are collected from the arterial or venous pressure, which at present is measured using invasive catheters inserted into arteries or veins. MAP, systolic pressure, and diastolic pressure are directly calculated from ART, as the mean, maximum, and minimum values respectively. The fact that each of them had an R^2 score greater than 0 for models trained on motion features indicates some level of predictability in blood pressure from perfusion motion.

B. Pixel Intensity Values

The outcome of the pixel intensity models are summarized in third column of Table 1. The pixel intensities of microvascular videos on their own are not useful for predicting vital signs, with only a few percentage points of variance being accounted for by only three out of 20 vital signs.

TABLE I
R^2 RESULTS OF VITAL PREDICTION

Vital Sign	Motion Feature R^2	Pixel Intensity R^2	Frequency R^2
ART	0.136	-0.015	-0.376
ECG	-0.039	-0.714	-1.091
CCO	0.160	-0.014	-0.325
CVP	0.100	0.048	-0.690
PAP	-0.276	-0.690	-0.348
Pleth	-1478	-1216	-1706
SvO2	0.192	0.034	-0.592
SpO2	-371.7	-383.1	-469.9
MAP	0.114	-0.022	-0.374
Sys	0.076	0.012	-0.459
Dia	0.126	-0.126	-0.298
PP	-0.230	-0.147	-0.476
HR	-0.020	-0.507	0.568
SV	-0.020	-0.416	-0.530
CO	-0.093	-0.673	0.406
HRV	-1.729	-2.502	-4.968
SPV	-0.806	-1.726	-0.450
SVV	-0.771	-1.171	-0.879
PPV	-0.582	-1.157	-1.012
PPV/SVV	-0.980	-1.421	-0.277

C. Pixel Intensity Frequencies

The outcome of the pixel intensity HR frequency models are summarized in the fourth column of Table 1. Frequencies outside the [0.5, 5] Hz region did not in any vitals correspond to a better than random R^2 score. Unsurprisingly, this range tuned to the heart rate showed the highest R^2 score for the heart rate, confirming past observations of heart rate in microvascular videos [13]. It also performed better than any other model at predicting cardiac output, a vital of interest due to the necessity of a partial anesthesia to measure it.

V. DISCUSSION

A. Value and Applicability

Invasiveness in surgical monitoring methods delays procedures and carries risks in its own right, so there is incentive to avoid invasive measurements in favor of other methods where possible. The model in this paper, trained on a controlled critical care experiment data set, indicates that several invasive vitals important in critical care are predictable from a purely visual data source. Blood pressure is amongst the most invasively measured, and in motion feature experiments we found evidence of predictability in venous (CVP, SVO2) and arterial (ART, MAP, Sys, Dia, CCO) pressure measures. Future work further investigating retrieving these signals visually would be clinically valuable to circumvent these more risky methods of data collection.

In frequency experiments, it should be noted that cardiac output, the volume of blood pumped by the heart per minute, is directly related to heart rate, so this predictive power can mostly be attributed to this model's success at identifying heart rate. This is also confirmed because this model does not predict stroke volume, which when multiplied by heart rate yields cardiac output.

Experiments on appearance features for heart rate prediction saw no change in performance between background and vessel pixels. There are several possible causes for this, including the heart rate causing physical disturbances to the microvascular monitoring camera. Alternatively, subsurface scattering of light under the skin medium may cause information to be diffused even to non vessel pixels.

B. Limitations and Future Work

This work was challenged by several limitations. Despite the critical care pig bleed dataset carefully controlling for many variables and offering a wealth of key information, it is at the time of writing almost seven years old, and is noisier and lower quality than more modern methods of microvascular video collection. It was also limited to 52 videos over 16 pigs. Finally, the videos themselves were not precisely timed with events, but rather taken during specific periods of the surgery (baseline, during bleed, after resuscitation), so synchronization between vitals and videos is imperfect. Using newer and more AI-ready datasets may improve results. Ideally this type of medical data would be less sporadically measured across subjects and surgeries, of higher quality, and better synchronized.

With regards to how the current dataset might be further utilized, only the frequency analysis examined the temporal aspect of the video data. Motion features compressed several hundred frames into a single histogram map. Future experimentation could examine several consecutive motion featurizations to better leverage temporal information.

Finally, vitals are valuable because they predict outcomes, so future work might investigate predicting outcomes directly from videos, skipping the intermediate step of vital reconstruction.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774, as well

as by a CMU Robotics Institute lab scholarship. Emma Erickson thanks the Auton lab for their immense support, particularly Dr. Robert Edman and Dr. Artur Dubrawski for their mentorship and guidance. She is also grateful to RISS program advisors Rachel Burcin and Dr. John Dolan for making the program possible in the summer of 2020. Finally she would like to thank her home university, the University of Illinois at Urbana-Champaign, for preparing her for this research.

REFERENCES

[1] "Nqf-endorsed measures for surgical procedures, 2015-2017," Final Report funded by the Department of Health and Human Services, National Quality Forum, p. 5, April 2017.

[2] M. E. Stokes, X. Ye, M. Shah, K. Mercaldi, M. W. Reynolds, M. F. Rupnow, and J. Hammond, "Impact of bleeding-related complications and/or blood product transfusions on hospital costs in inpatient surgical patients," *BMC Health Serv Research*, vol. 11, May. 2011.

[3] M. R. Pinsky, A. Wertz, and A. D. G. Clermont, "Parsimony of hemodynamic monitoring data sufficient for the detection of hemorrhage," *Anesthesia Analgesia*, vol. 130, May 2020.

[4] D. D. Backer and A. Durand, "Monitoring the microcirculation in critically ill patients," *Best Practice Research Clinical Anaesthesiology*, vol. 28, Dec. 2014.

[5] S. Trzeciak, R. P. Dellinger, J. E. Parrillo, M. G. J. Bajaj, R. C. A. Nicole L Abate, S. Colilla, S. Zanotti, and S. M. Hollenberg, "Early microcirculatory perfusion derangements in patients with severe sepsis and septic shock: relationship to hemodynamics, oxygen transport, and survival," *Ann Emerg Med*, vol. 49, Jan. 2007.

[6] A. Spanos, S. Jhanji, A. Vivian-Smith, T. Harris, and R. M. Pearse, "Early microvascular changes in sepsis and severe sepsis," *Shock*, vol. 33, Apr. 2010.

[7] M. Hilty, P. Guerci, Y. Ince, F. Toraman, and C. Ince, "Microtools enables automated quantification of capillary density and red blood cell velocity in handheld vital microscopy," *Communications Biology*, vol. 2, Jun. 2019.

[8] S. López, G. Suarez, D. Jungreis, I. Obeid, and J. Picone, "Automated identification of abnormal adult eegs," *2015 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, Dec. 2015.

[9] S. Roy, I. Kiral-Kornek, and S. Harrer, "Chrononet: A deep recurrent neural network for abnormal eeg identification," *Artificial Intelligence in Medicine*, May 2019.

[10] I. Ocak, A. Kara, and C. Ince, "Monitoring microcirculation," *Best Practice Research Clinical Anaesthesiology*, Dec. 2016.

[11] G. Guven, M.P.Hilty, and C. Ince, "Microcirculation: Physiology, pathophysiology, and clinical application," *Blood Purification*, Feb. 2020.

[12] C. Liu, H. Gomez, S. Narasimhan, A. Dubrawski, M. R. Pinsky, and B. Zuckerbraun, "Real-time micro-vascular video analysis," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[13] C. Liu, "Vision with small baselines," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2020.

Planning using Physics-Based Simulation for Contact-Rich Assembly Tasks with Environmental Uncertainty

Ethan Fahnestock[1], Shivam Vats[2], and Maxim Likhachev[2]

***Abstract*— The task of parts assembly is a contact-rich manipulation task that is sensitive to environmental uncertainty, motivating the traditionally structured and controlled factory environment. However, assembly tasks are common in unstructured environments where robots do not know exact locations of objects. Recent approaches have demonstrated that policies to solve assembly tasks with environmental uncertainty can be found, but at a potentially prohibitive computational cost that makes these approaches slow to adapt to new problems. In this paper, we propose a method to generate robust policies for parts assembly in an attempt to address this limitation. We evaluate this method in simulation on the "*board insertion task*". We quantify performance of the proposed method against a metric planner baseline and demonstrate improved task completion rates at lower environmental uncertainties.**

***Index Terms*— assembly, physics-based simulation, contact-rich manipulation**

Fig. 1. An example of a household contact-rich assembly task

I. INTRODUCTION

Robot manipulators are used for assembly tasks with increasing frequency to improve efficiency of production lines across industries in structured factory environments. However, assembly tasks are not unique to factories or assembly lines, and frequently occur in unstructured environments. Any manipulation task that requires arranging two or more items within relative poses of each other can be classified as an assembly task. These tasks are typically contact rich, which makes solving these manipulation tasks sensitive to environmental uncertainty.

For example, take the jar and lid shown in Figure 1. This common household task of screwing on a lid to a jar demonstrates the contact-rich nature of assembly tasks, as completing the task necessarily involves contact between the lid and the jar. This example also illustrates the task's sensitivity to environmental uncertainty. For instance, if the lid is slightly off-center because of error in the pose estimation of the jar, it will not screw on and the assembly task will fail.

This sensitivity to environmental uncertainty motivates robot assembly to traditionally occur in highly structured environments. Industrial robots, like those used on car assembly lines, are expensive and typically rely on their precision, repeatability, and low noise environment to complete steps in pre-programmed sequences designed to complete the specific assembly task at hand. These pre-programmed sequences are fragile and cannot adapt to even small changes in the environment.

As robots are deployed in less structured environments where they must plan in novel surroundings they require new methods of manipulation that can better handle the uncertainty inherited from perception to successfully complete assembly tasks. Further, these methods need to be able to handle contact, unlike traditional motion planners that avoid contact with the world. Contact complicates the planning problem, as contact forces are non-linear and predicting their effects on a robot's action is difficult. Additionally, manipulators must not damage themselves, or the objects they are manipulating, while executing a solution generated by a planner involving contact. Introducing compliance in the manipulator is a popular technique to make interactions between the robot and the world safer by limiting the maximum contact force. It allows manipulators to adapt to external forces (i.e. can be pushed around) instead of executing a sequence of positional waypoints in a non-back-drivable manner. Compliant manipulators can either be actively compliant, where the compliance is implemented in software controllers, or passively compliant, where the manipulator itself is constructed with elastic elements that can absorb energy [1]. In this work, we aim to utilize active compliance to support contact during manipulation.

Although contact introduces complications into planning, it can also be exploited to learn about a robot's environment. Humans utilize contact daily to complete precise assembly tasks that would be difficult otherwise. For example, while attempting to use a key in the dark, a person may first introduce contact between the key and the lock to localize features before trying to insert the key [2].

[1] Ethan Fahnestock is with the Hajim School of Engineering and Applied Sciences, University of Rochester, Rochester, NY 14627 efahnest@u.rochester.edu

[2] Shivam Vats and Maxim Likhachev are with the School of Computer Science, Carnagie Mellon University, Pittsburgh, PA 15213 {svats, mlikhach}@andrew.cmu.edu

Approaches have demonstrated that contact can be utilized by a robot manipulator to generate more robust solutions to assembly tasks, but these approaches either execute pre-programmed action sequences, or if they do plan, usually have runtime requirements that limit their use to scenarios where the robot has extended periods of time to plan when faced with a novel environment or manipulation task. When operating in unstructured environments, the specifics of a manipulation tasks can change on a timescale too rapid for these approaches to keep up.

In this work we aim to develop a planner that addresses this issue while leveraging contact to create plans robust to environmental uncertainty. We utilize parallelized physics-based simulation [3] and impedance-based control to incorporate contact in planning. We demonstrate our algorithm in simulation on the "*board insertion task*" and compare our approach against a metric planner baseline.

The structure of this paper is as follows: in Section II we review related works, Section III formalizes the problem and our proposed method, Section IV describes our experimental evaluation which produces the results found in Section V. Finally, we conclude in Section VI with a discussion of the results and future directions for this work.

II. RELATED WORKS

Many approaches have utilized contact to their advantage in manipulation tasks. Contact in [2] is used in programmed routines to assist visual sensors in localizing objects important for manipulation to complete tasks like key-insertion and picking up a screwdriver from a table.

Other approaches that focus specifically on parts assembly also utilize contact. For example, [4] treats parts assembly as a planning under uncertainty problem and utilizes contact while planning actions for a compliant manipulator by formulating the problem as a Markov Decision Process (MDP) with uncertainty in action execution. Like our work, this approach also makes use of physics-based simulation and impedance-based control. However, this work does not consider environmental uncertainty, only uncertainty in the robot's action.

In recent years numerous approaches have combined machine learning based approaches with impedance control for manipulation tasks, some involving contact and assembly. [5] demonstrates that the gain schedule (or stiffness of the controller over time) for an impedance controller can be learned to complete manipulation tasks involving contact while decreasing stiffness when it is not required, making operation safer.

In [6], a high-level policy is learned through reinforcement learning (RL) with contact states as context to chain low-level skills in a way that enables a manipulator to attempt to recover when an error is made. Like this approach, ours uses hand-designed "skills", or motion primitives, as well as executes actions with an impedance controller to achieve compliance.

Most relevant to our work in the learning space is [7]. This work models the problem of parts assembly with environmental uncertainty as a Partially-Observable Markov Decision Process (POMDP), modeling environmental uncertainty with a particle filter from which a Gaussian Mixture Model (GMM) is extracted. An RL agent operates on this compressed representation of the belief space to switch between goal-directed impedance controllers.

A limiting factor of POMDP-based approaches is that they don't scale with the dimensionality of the configuration space. [8] introduces a method that avoids this scalability issue by only considering reachable contact events and discretizing the state space during planning. This method utilizes stochastic Rapidly-exploring Random Tree (RRT) planning and contact feedback from sensors on the end effector to plan and execute contingencies based on contact states. Our work differs in our use of deterministic sampling methods.

Most similar to our work in this respect is [9], which uses deterministic sampling in planning for parts assembly with environmental uncertainty. Our work differs in our use of impedance-based control.

III. TECHNICAL APPROACH

A. Problem Definition and Assumptions

We aim to find a sequence of actions $A = \{a_1, a_2, ..., a_n\}$ that when executed by a manipulator achieve a goal relative pose $\mathbf{p_g} \in \text{SE(3)}$ between two objects involved in an assembly task within some tolerance $\delta\mathbf{p_g}$. We assume knowledge of the object's geometry and physical properties to enable physics-based simulation of the object, and accurate state feedback from the manipulator to perform the task space impedance control. Finally, we assume knowledge of the expected pose of the assembly object not held by the robot's gripper during the task.

B. Overview

Our approach first constructs a finite action space for the robot using a set of hand-designed controllers. These controllers are then used to build a graph $\mathcal{G}$, the nodes of which represent a state in the robot's configuration space. A traditional A* graph search [10] with an inflated heuristic is used to search and construct $\mathcal{G}$. During node expansion, the actions are applied to the parent node's state and forward-simulated in a physics-based simulator. The states resulting from these actions, if valid, are returned as successors to the parent state. Search continues until the relative transform between assembly objects lies within $\delta\mathbf{p_g}$, or a time limit is exceeded. The details of this approach are described in the following sections.

C. State Lattice with Controllers

We utilize State Lattice with Controllers (SLC) [11] to construct our graph in configuration space. Instead of traditional motion primitives that modify the robot's configuration in a fixed way, the employed motion primitives are controllers that operate on the current robot's state. As described in the previous section, these controllers are then simulated

Algorithm 1: GetSuccs

Input: Parent state s_p, vector of motion primitives M
Output: A set of successor states S

```
1  S = ∅;
2  for envidx, m_p ∈ enumerate(M) do
3  |   SetRobotState(envidx, s_p);
4  |   G_i ← ApplyCartesianTransform(s_p, m_p);
5  |   SetAttractorPose(envidx, G_i);
6  end
7  while False ∈ {x.complete | x ∈ M} do
8  |   for envidx, m_p ∈ enumerate(M) do
9  |   |   complete ← CheckComplete(envidx, m_p);
10 |   |   if complete and not m_p.complete then
11 |   |   |   if CheckValidConfiguration(envidx) then
12 |   |   |   |   s← GetRobotState(envidx);
13 |   |   |   |   S.append(s);
14 |   |   |   end
15 |   |   |   m_p.complete ← complete;
16 |   |   end
17 |   end
18 |   StepSimulation();
19 end
20 return S
```

to produce successor states. This is illustrated in Algorithm 1.

The controllers are made up of two components: a termination condition and a relative transform. The termination condition checks for completion of the action after each simulation step. The relative transform is applied to the end effector pose at the beginning of an action to generate a target pose in $SE(3)$ that is fed to a task space impedance controller, described in the following section.

D. Task Space Impedance Control

Like [12], we employ a task space impedance controller with attractor states. This controller achieves active compliance and allows for better integration of contact into planning. The controller follows a spring-damper model, illustrated in Equation 1 [13]. Here, $\mathbf{M_d}$ is the mass matrix of the manipulator, $\ddot{\tilde{\mathbf{x}}}$ is the operational space error, $\mathbf{K_D}$ is the damping matrix, $\mathbf{K_P}$ is the stiffness matrix, and $\mathbf{h_e^d}$ are the contact forces exerted on the environment by the end effector. The attractor pose that is used to define $\ddot{\tilde{\mathbf{x}}}$ creates a spring-like attraction between the end effector and the attractor state that "pulls" the end effector until an equilibrium position is reached.

$$\mathbf{M_d}\ddot{\tilde{\mathbf{x}}} + \mathbf{K_D}\dot{\tilde{\mathbf{x}}} + \mathbf{K_P}\tilde{\mathbf{x}} = \mathbf{h_e^d} \tag{1}$$

E. Termination Conditions

Two termination conditions are explored in this work. The first monitors the maximum linear and angular end effect velocity components, and after a brief period to allow ramp-up, terminates the action after the maximum linear and angular velocity components drop below their corresponding thresholds.

The second monitors for the detection of contact between the object and the environment. In our experiments, contact information was extracted from the simulator, while in the real world this could be estimated using a force-torque sensor. Unlike actions with velocity termination conditions, where the relative transform is used to set the attractor state once at the beginning of the action, actions with contact termination conditions update their attractor state after every simulation step. This allows these actions to "move until contact" without setting distant attractor states that would cause the impedance controller to generate large forces and move quickly.

All actions, regardless of their termination condition, are terminated after the number of simulation steps exceeds a set threshold.

Once a termination condition is reached in simulation, the attractor state is removed and the robot's configuration is immediately captured and eventually returned as a successor state to a node expansion if valid.

F. Valid Configuration Checking

The simulation of controllers in certain scenarios can result in invalid states where the object held by the robot's gripper is displaced through interaction with its environment. To prevent these states from influencing search, they are discarded if determined invalid, as outlined in Algorithm 1 on line 11. The CheckValidConfiguration function checks the relative transform between the robot's gripper and the object it is holding, and discards the state if the relative pose has shifted translationally or rotationally from its initial value before controller simulation by more than set thresholds for each.

G. Physics-Based Simulation for Node Expansion

Accounting for contact dynamics using physics-based simulation comes at a high computational cost. To mitigate this, we utilize Nvidia's Isaac Gym [3], a GPU-based physics simulator that can efficiently parallelize the simulation of individual environments. We simulate environments for each controller to parallelize the collection of successor states as shown in Algorithm 1 and visualized in Figure 2.

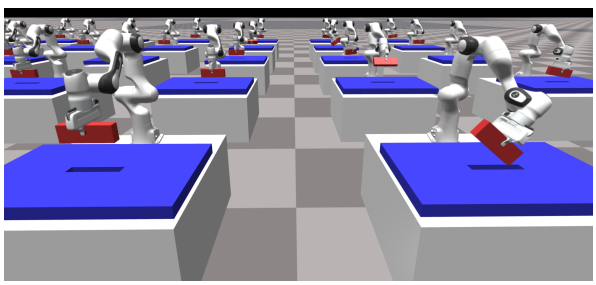


Fig. 2. A depiction of parallel motion primitive simulation for the "*board insertion task*" described in Section IV

H. Heuristic and Edge Cost

To compute a heuristic for the A* search we took the norm of the differences between the XYZ positions of the current state and goal state, and summed it in a weighted fashion with the angle between each state's orientation. For this work, we weighted the cartesian offset (in meters) to angle (in radians) at a ratio of 10:3. A fixed edge cost was used during planning.

I. Solution Execution

If a solution is found by the planner, the sequence of actions A is constructed from the ordered sequence of motion primitives extracted from the solution path's edges. A is executed in-sequence in simulation.

J. Metric Planner Baseline

We compare our results against a traditional metric planner baseline that avoids collisions and executes actions with a non-compliant controller. This planner does not use physics-based simulation during node expansion, instead using a set of metric motion primitives that directly modify the joint state of the robot. These metric primitives add or subtract a fixed value from a DOF of the manipulator. Additionally a "snapping" motion primitive is applied, using inverse kinematics to find joint states resulting in the goal end effector pose. The planner uses a spheres model, consisting of of the robot and board, as well as an occupancy grid representation of the slot during search to check for collisions in the child states generated in an expansion, discarding them if they are invalid. These collision models can be seen in Figure 3. The metric planner was implemented using the Search-Based Motion Planning Library [14].

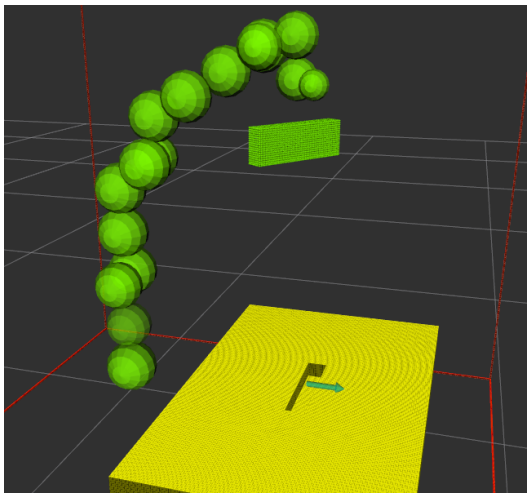


Fig. 3. Visualization of the occupancy grid and spheres collision model used by the metric planner

As physics-based simulation was not used, the board was fixed to the end effector. The same euclidean distance heuristic with orientation was used for the metric planner as described in III-H but with a ratio of cartesian offset to angle of 10:1. The vertical component of the cartesian displacement was also reduced by half before computing the offset to prioritize errors in the plane of the table. A constant edge cost was applied in an identical manner to the SLC method.

IV. EXPERIMENTAL SETUP

We evaluate our planner in simulation on the *"board insertion"* assembly task. The environment for this task can be seen in Figure 4 with the board in red and slot in blue. The board starts in the robot's gripper. There is a 0.5mm clearance on all sides between the board and slot. Although with no environmental uncertainty this assembly task can be solved without contact, we consider this a contact rich assembly task because the introduction of environmental uncertainty above clearance levels makes solutions that do not involve contact improbable.

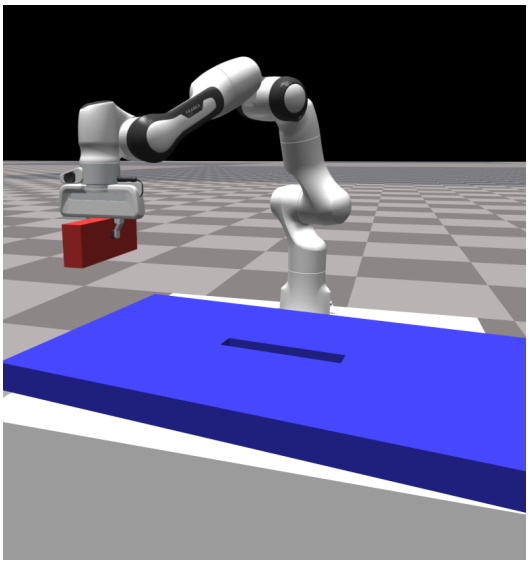


Fig. 4. An image of the simulation environment for the "*board insertion task*"

We simulate a FRANKA EMIKA Panda 7DOF manipulator with a 2-finger prismatic gripper. For the 7 planning joints (excluding the gripper), a stiffness of $2000 Nm/rad$ and damping of 100 $Nm/(rad/s)$ are applied, with an attractor stiffness of $100 N/m$ and damping of $250 N/(m/s)$.

Experiments for all planners are performed on the same machine running Ubuntu 16.04 with an Nvidia GeFORCE 1050ti GPU, 8GB of RAM, and a 4 core processor.

In total, 19 motion primitives are used by the SLC approach. One "move until contact" primitive was used that moves the gripper down towards the slot. The remaining 18 primitives used end effector velocity termination conditions. End effector linear and angular velocity thresholds of $0.01 m/s$ and $0.01 rad/s$ are used. Twelve of these primitives have relative transforms that apply $\pm 0.1m$ and $\pm 0.4m$ offsets to each of the three cartesian DOFs. The remaining six primitives contain relative transforms that apply $\pm 0.3 rad$ offsets to roll, pitch, and yaw. The simulation step threshold for all actions is set to 300 steps, corresponding to 3 seconds of simulated time at 0.01 seconds per step.

For checking the validity of successor states for the SLC planner, the linear displacement threshold is set to $1mm$ and angular displacement threshold is set to $0.1rad$.

The metric planner used 23 motion primitives. Fourteen of these applied $\pm 0.02rad$ to each DOF, eight applied $\pm 0.14rad$ to the first four DOFs (starting at the base of the arm), and the last motion primitive was a "snapping" primitive active only when the end effector was within 0.2 meters of the goal.

As the metric planner did not utilize impedance control, contact with the environment could be damaging to the robot and its surroundings. Thus, solutions generated by the metric planner were deemed invalid if contact was detected between the robot and its surroundings during path execution. To determine success rates for the metric planner noise was added to the slot position. Every state in the path generated by the planner for the original slot position was then checked for collisions, and the trial was identified as successful if no state in the path resulted in collision.

To compare the considered approaches, we gather statistics on planner performance and then execute the generated solutions in 300 environments with added gaussian noise to four of the six degrees of freedom (X, Y, Z and yaw) of the slot. We perform this evaluation with the standard deviations of these distributions varying between 1 mm/mrad to 5mm/mrad in 1mm/mrad increments.

V. RESULTS

Figure 5 compares the success rates of the baseline and proposed approach as environmental uncertainty is increased. Planning statistics for both approaches can be seen in Table I.

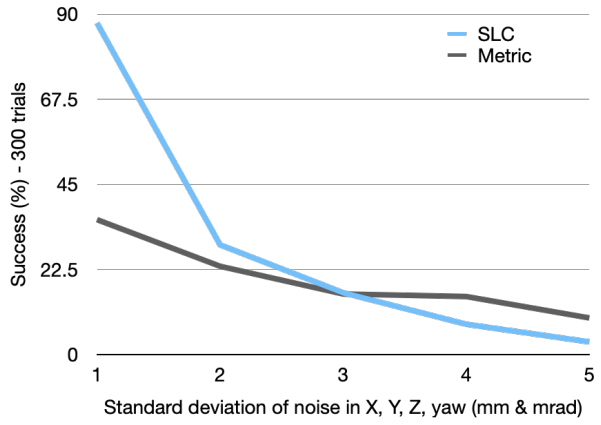


Fig. 5. Performance of planners with increasing environmental uncertainty

As can be seen in Table I, the cost of physics-based simulation is obvious in the planning time differences between approaches. Our proposed method successfully completes the assembly task at twice the rate of the metric planner with low (1mm) variance in environmental noise. This quickly changes with increased environmental uncertainty, with our proposed approach being surpassed by the metric planner above standard deviations of 3mm.

TABLE I
PLANNER PERFORMANCE

Planner	Metric	SLC
Planning time (s)	43	388
Node expansions	3040	14
Time per node expansion (s)	0.014	27.7
Solution path size (actions)	137	14

VI. CONCLUSIONS

This preliminary work was able to demonstrate that the use of physics-based simulation and impedance control alone, without explicitly considering environmental uncertainty, can increase the success rate of assembly tasks with limited environmental uncertainty over a metric planner baseline in exchange for a respectable additional computational cost.

Planning time still sits as a clear limitation of the proposed approach. With its current design, node expansion is as slow as its slowest controller. If the proposed approach could be modified to return successor states asynchronously, this constraint could be relaxed as search could continue while integrating successor states as they were produced. This may reduce the time cost of search, and make this approach viable for faster-changing environments.

An additional limitation of the proposed approach is its reliance on hand-designed controllers, such as the vertical "move until contact" controller, that while beneficial for board insertion, is unlikely to generalize well and generate useful actions for other classes of assembly tasks like key insertion. Future work should look to address this issue and evaluate the performance of the proposed method across a variety of manipulation tasks.

The method proposed in this work does not explicitly reason about environmental uncertainty, and relies on impedance-based control to implicitly adapt when objects are not at their expected positions. An extension to this method is being explored that samples environmental configurations from a noise model during planning and biases search towards actions that reduce the variance in resulting successor states with respect to the other assembly object. It is hypothesized that this approach will generate solutions more robust to environmental uncertainty, but at the cost of longer planning times.

There also remains significant work in evaluating the proposed approach on a physical robot. Tuning of simulator parameters is likely required to get plans found in simulation to translate effectively to a physical platform. Additionally, contact detection using a force-torque sensor would have to be implemented for contact termination conditions.

Another area of potential future work is using the solutions generated with the proposed method to seed training for RL agents attempting to learn policies to solve contact-rich assembly tasks. This may benefit the agents by reducing required training time compared to agents that start with random policies.

ACKNOWLEDGMENT

The first author would like to acknowledge the organizers of the Robotics Institute Summer Scholars program, Rachel Burcin and Dr. John Dolan, for their tremendous support and hard work. This opportunity would not have been possible without them, especially given the unconventional circumstances this summer. The first author would also like to thank Shivam Vats and Dr. Maxim Likhachev for their mentorship and guidance. This material is based upon work supported by the National Science Foundation under Grant No. 1659774. This work was also supported through the CMU RI Lab Scholarship.

REFERENCES

[1] A. A. Schäffer, O. Eiberger, M. Grebenstein, S. Haddadin, C. Ott, T. Wimböck, S. Wolf, and G. Hirzinger, "Soft robotics, from torque feedback-controlled lightweight robots to intrinsically compliant systems," *IEEE Robotics and Automation Magazine*, vol. 15, no. 3, pp. 20–30, 2008.

[2] N. Hudson, T. Howard, J. Ma, A. Jain, M. Bajracharya, S. Myint, C. Kuo, L. Matthies, P. Backes, P. Hebert, *et al.*, "End-to-end dexterous manipulation with deliberate interactive estimation," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2371–2378.

[3] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," *arXiv preprint arXiv:1810.05762*, 2018.

[4] C. Guan, W. Vega-Brown, and N. Roy, "Efficient planning for near-optimal compliant manipulation leveraging environmental contact," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 215–222.

[5] J. Buchli, E. Theodorou, F. Stulp, and S. Schaal, "Variable impedance control a reinforcement learning approach," *Robotics: Science and Systems VI*, pp. 153–160, 2011.

[6] A. S. Wang and O. Kroemer, "Learning robust manipulation strategies with multimodal state transition models and recovery heuristics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1309–1315.

[7] F. Wirnshofer, P. S. Schmitt, G. v. Wichert, and W. Burgard, "Controlling contact-rich manipulation under partial observability," *Robotics: Science and Systems XVI*, 2020.

[8] E. Páll, A. Sieverling, and O. Brock, "Contingent contact-based motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6615–6621.

[9] S.-K. Kim and M. Likhachev, "Parts assembly planning under uncertainty with simulation-aided physical reasoning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4074–4081.

[10] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: https://doi.org/10.1109/tssc.1968.300136

[11] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev, "State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 258–265.

[12] M. S. Saleem and M. Likhachev, "Planning with selective physics-based simulation for manipulation among movable objects," *arXiv preprint arXiv:2003.06743*, 2020.

[13] S. Bruno, S. Lorenzo, V. Luigi, and O. Giuseppe, "Robotics: modelling, planning and control," 2010.

[14] "Search-based motion planning library," https://github.com/aurone/smpl.

Planning for Real-time Blocking Flying Objects with a Redundant Manipulator

Zeyuan Feng[1], Ramkumar Natarajan[2] and Maxim Likhachev[2]

***Abstract*— We consider the task of autonomously blocking flying objects for "robotcops" who are capable of guarding in cities. Such ability could be further applied to robots in partial-controlled scenarios, such as building sites, for self-protection. There already exist various algorithms which generate motion plans for high-dimensional manipulators. However, none of them can be applied in our domain due to two main challenges. As robots need to block objects in a short limited time, the motion planner is required to plan in real-time to get a time-optimal or suboptimal plan. In addition, as objects are often thrown from distance, robot's perception system would gradually get better estimation when an object is getting closer. Consequently, the proposed planner should be able to adjust its motion plans when the estimation is updated. We present a new approach to meet those requirements by using offline auxiliary information. In simulation, We validate the performance of the proposed motion planner under different cases.**

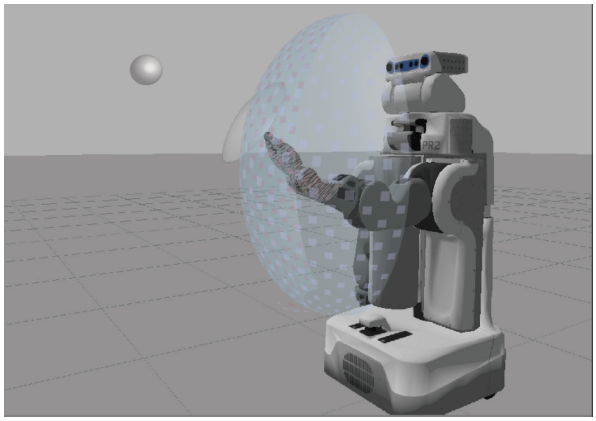


Fig. 1. The task is to generate a motion plan for manipulator to shield flying objects in a predefined surface around robot body

I. INTRODUCTION

In the modern times robots are expected to be deployed in unknown or complex scenarios to decrease workload and risk of human workers. The ability of self-protection is essential for robot in such real-world tasks since it prevents robots from damage to dramatically trim unnecessary cost of money and time. Such ability is also necessary for those robots which involve training in physical world since it prevents naive actions in early learning phase from hurting robot bodies. Existing work on robot self-protection focused on different aspects of the problem ranging from protecting fragile robot actuation [1], mechanical overload protection [2] and fall down protection [3]. However, using robot manipulator to block flying object, which is important for robot body protection, is still understudied. In this work, we consider the problem of motion planning for shielding off flying objects in a predefined surface in front of robots. Specifically, a robot will move an attached shield by its manipulator to a goal pose, which lies on a predefined surface, to block objects flying toward it.

To the best of our knowledge, there is no previous work on blocking object with manipulator. Although our task is similar with motion planning for reaching process of a pickup-object mission, the existing planners could be applied to our task directly. [4] use a kinodynamic motion planner to smoothly reach the moving objects. However, this planner cannot be used online while an object in flying imposes the requirement of short planning and reaching time. Apart from this, our problem can be modeled as a Moving Target Search problem as well. Existing approaches [5], [6] consider the case of heuristic search in 2D where an agent (hunter) is to catch a moving target (prey). Nevertheless, they are too computationally expensive to be applied to our task due to our high-dimensional property. [7] presented an efficient CNN architecture for real-time manipulator grasping. However, such learning approach is expensive to be implemented when various shields are attached.

The shielding process highly relies on quality detection and localization of flying objects since we use the object's pose to compute its landing pose, namely our goal state, in the predefined surface. Unfortunately, the initial perception estimates of the object's pose are inevitably inaccurate due to perception noise. What is worse, as the object is far from robot initially, a slight error of the object's pose will lead to a large error for its landing pose. A proper filter can help to reduce the detection error gradually. Thus, the landing point estimation becomes fairly accurate only if the object moves closer as well as the filter error decreases. However, if the robot waits too long to get an accurate estimate, the delay in starting plan execution could cause the shield to miss the object. The robot therefore should start executing a plan computed for the initial goal pose. In addition, when the robot gets better estimation for the goal states, it should repeatedly replan for the new goals. For every replanning query, the time window for shielding shrinks. This makes the time for each planning step even less. Since planning problem is high-dimensional and requires collision avoidance as well, it's infeasible to purely plan online. [8] proposed

[1]Zeyuan Feng is with the Department of Electronic Information Engineering, The Chinese University of Hong Kong, Shenzhen, China zeyuanfeng@link.cuhk.edu.cn

[2]Ramkumar Natarajan and Maxim Likhachev are with the School of Computer Science, Carnagie Mellon University, Pittsburgh, PA 15213 {rnataraj,mlikhach}@andrew.cmu.edu

a planner that leverages offline preprocessing to provide bounds on the planning time when the planner is invoked online. However, this approach relies on the assumption that there's a replan cutoff time. In our task, the time for a object to hit the surface depends on the initial pose and velocity of the object. Consequently, for every case the robot should reach its goal state as soon as possible and we cannot assume a fixed cutoff time for it. This suggests the experience-compressing technique in [8] cannot be applied directly in our task. To the best of our knowledge, there is no planner satisfies our specific task.

In this work, we only consider the real-time challenge of planning and blocking. Our planning algorithm based on a provable real-time planner [9] that leverages compressed offline experience for only blocking. This real-time planner is used in pre-processing phase to compute and store important paths to so-called "attractors". An attractor is matched to a "subregions" where we can find a collision-free path from any state inside to the attractor by greedy search. We finely define the problem and present how to apply this planner to a goal region that the shield is on a curved surface. We complete the query algorithm that finds goal states and generates motion plans by combining the pre-computed path from initial state to the attractor and greedy path from attractor to goal state. The effectiveness of our algorithm is demonstrated in simulation on a PR2 robot.

II. PROBLEM DEFINITION

The task presented in this paper is to generate collision-free motion plans for a robot to block objects flying toward it for self-protection. We define the world state $\boldsymbol{W}$ to be comprised of a robot, an attached shield and an object $\boldsymbol{O}$. The shield is a cylinder with negligible height h_s and a radius r_s. At every point of time, there is at most one object in the scene attacking the front side of the robot. The pose of the object g_o are detected by robot's perception system.

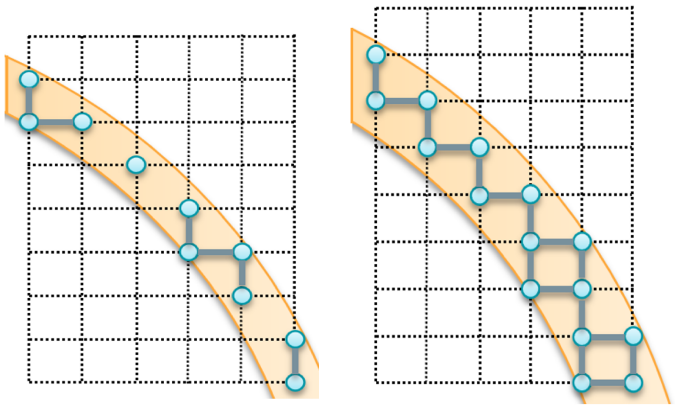
Our planner $\boldsymbol{P}$ takes g_o as input. If $\boldsymbol{O}$ is going to hit the robot body, its landing pose $g_l = [p_l, v_l]$, at which $\boldsymbol{O}$ will land on a predefined surface, will be determined. Then $\boldsymbol{P}$ will output a collision-free motion plan for the robot's manipulator $\boldsymbol{R}$ to move to a valid goal configuration s_{goal} for defense. In fact, the robot can use its shield to block an object at different positions and orientations thanks to the shield's area. Namely, at s_{goal}, the position of the shield p_s can be slightly deviated from p_l and the normal vector of shield surface v_s can be slightly unparalleled to v_l as long as they guarantee successful blocking. Hence, given an g, $\boldsymbol{P}$ probably generates a set of goal configurations. It will try them out until successfully query a plan for $\boldsymbol{R}$ to execute. Apart from this, we assume $\boldsymbol{R}$ starts from an fixed initial configuration s_{home} in each blocking process. s_{home} is a predefined configuration where $\boldsymbol{R}$ can reach any s_{goal} in goal region $\boldsymbol{G}$ within a relatively fixed and small amount of time. We emphasize that all poses in this paper are represented in robot's body frame, which can be directly measured by attached sensors.

A. Surface Definition

The surface should be defined to offer sufficient protection to the front of the robot. Compared to a plane, a curved surface is more reasonable to be adopted. Further considering the simplicity, we define the surface as a spherical cap.

$$\begin{cases} (x - x_{org})^2 + (y - y_{org})^2 + (z - z_{org})^2 = r^2 \\ Angle((x - x_{org}, y - y_{org}, z - z_{org}), (1, 0, 0)) < \alpha \end{cases} \quad (1)$$

Where x axis is positive forward and negative backward while z axis is positive upward and negative downward. Typically $x_{org} < 0$, $y_{org} = 0$ and $z_{org} = \frac{h_{robot}}{2}$.



(a) (b)

Fig. 2. The orange region is a cross section of the surface. Each blue dot in the figure represents several goal configurations of $\boldsymbol{R}$ where shield's positions are the same with dot's position in the cross section and shield's orientations are different. Blue lines state if there are any two configurations in two blue dots connected. (a) When $\epsilon < \frac{\sqrt{2}\Delta_{max}}{2}$, we cannot implicitly build the graph of goal region since the graph of goal region is not connected. (b) When $\epsilon > \frac{\sqrt{2}\Delta_{max}}{2}$, the graph of goal region is guaranteed to be strong connected.

B. Goal Region Definition

We discretize the manipulator's configuration space $\boldsymbol{C}$ into a state lattice $\boldsymbol{S}$. In addition, each state is connected to its successors and predecessors by a set of motion primitives. The motion planning problem can therefore be transferred to a graph search problem. That is, the planner can search a path between two states on the graph for motion planning. However, the constructed graph may not be strong connected due to the curved surface setting, which means we may not find a path from one state to another, unless we enlarge $\boldsymbol{G}$ to be the set of s_{goal} at which distance between shield position and the surface is under a threshold $\epsilon > \frac{\sqrt{2}\Delta_{max}}{2}$), where Δ_{max} is the maximum displacement of motion primitives (see Fig. 2). The region of shield's goal positions is actually several discrete rings on different planes. Lastly, the set of s_{goal} within $\boldsymbol{C}$ is denoted by $G_S = C \cap G$

III. METHOD

A provably indefinite-horizon real-time planning algorithm was proposed in [9]. Based on the same thought of paths-compression mechanism, our algorithm framework includes

an offline preprocessing phase and an online object-blocking phase. The planner generates paths to the goal region and stores them in computer memory in the preprocessing phase while queries paths from memory to speedup planning process in the object-blocking phase.

A. Preprocessing Phase

We employ the preprocessing algorithm, which is presented in details in [9]. The preprocessing phase takes as input the initial configuration s_{home}, the surface definition and a conventional motion planner, and outputs a set of subregions and the corresponding library of paths from s_{home} to each $s^i_{attractor}$. Before explaining the algorithm, we introduce two definitions.

- A configuration is valid if the manipulator does not collide with itself and the robot body under this configuration.
- A configuration is invalid if the manipulator collides with itself or the robot body under this configuration.

The algorithm maintains a set of frontier valid states V and a set of frontier invalid states I to help finding uncovered valid states and invalid states, respectively. Both V and I are empty at the beginning. The preprocessing phase initiates with sampling a state in G_S and push it into V. Then it covers the whole G_S with subregions by repeating following pipeline until both V and I get empty.

- Find a state s not covered by any subregion as i^{th} attractor $s^i_{attractor}$:
  - If V is not empty, repeatedly pop the first state from V until the state s is uncovered yet. Set s as $s^i_{attractor}$.
  - Otherwise, "go through" invalid regions to find a uncovered valid state and push it into V. Then jump to the next round.
- Compute a hyperball subregion G_i center at $s^i_{attractor}$ with a radius r_i by the algorithm $ReachabilitySearch$ to cover surrounding states.
- Generate a path Π_i for $\boldsymbol{R}$ to reach the $s^i_{attractor}$ from s_{home}. Store the path, attractor number and the subregion's radius.
- Get corresponding frontier states, which are just outside G_i, and push them into V and I according to whether valid or not.

When the iteration stops, all attractors with their corresponding paths and radius are stored in computer memory. Some further explanation may help understanding the algorithm. When we "go through" invalid regions, invalid states will be popped out after checking in order to make sure I ends up to be empty. G_i includes and only includes states that have a heuristic value $h(s, s^i_{attractor}) < r_i$. A collision free path from any state to the attractor of its subregion can be found by a simple greedy search.

B. Object-Blocking Phase

In Object-blocking phase, the robot keeps sensing the environment. When a new object is detected, the algorithm calls the procedure $BlockObject$ once. The precedure takes $g_o = [p_o, v_o]$ as input. It starts by predicting g_l and produce a stack of manipulator's goal configurations G_s that contains multiple choices for blocking. Poping the first s_{goal} from G_s, it then finds an attractor to which the heuristic value of goal configuration $h(s_{goal}, s^i_{attractor}) < r_i$. If there is no such attractor or s_{goal} is invalid, the planner will recover by popping next s_{goal} to query unless the stack gets empty. After getting the corresponding attractor $s^i_{attractor}$, a greedy search is employed to give the path from s_{goal} to $s^i_{attractor}$. We reverse the greedy path and append it into the pre-computed path π_i to get the completed path for $\boldsymbol{R}$ to execute.

Algorithm 1 BlockObject

```
1:  procedure BLOCKOBJECT(p, v)    ▷ Pose of the object
2:      G_c = GetGoalConfigurations(p, v)
3:      π ← ∅
4:      while !G_c.empty() & π = ∅ do
5:          s_goal = G_c.Pop()
6:          for each R_i ∈ R do
7:              if h(s_goal, s_i) < r_i then
8:                  π ← GetGreedyPath(s_goal, s_i)
9:                  π ← π_i.Append(π.Reverse())
10:                 break
11:         if π ≠ ∅ then
12:             Execute(π)
```

1) Goal state prediction: In this preliminary work, the object is modeled as a mass point without any aerodynamics characteristics. Hence, the its trajectory is a parabola

$$\begin{cases} x = x_0 + v_x t \\ y = y_0 + v_y t \\ z = z_0 + v_z t - \frac{gt^2}{2} \end{cases} \tag{2}$$

Combine the trajectory with the surface representation (1), we get a quartic equation of t

$$at^4 + bt^3 + ct^2 + dt + e = r^2 \tag{3}$$

where

$$\begin{aligned} a &= \frac{1}{4}g^2 \\ b &= -gv_z \\ c &= v_x^2 + v_y^2 + v_z^2 - gz_0 + z_{org}g \\ d &= 2(x_0v_x - x_{org}v_x + y_0v_y - y_{org}v_y + z_0v_z - z_{org}v_z) \\ e &= (x_0 - x_{org})^2 + (y_0 - y_{org})^2 + (z_0 - z_{org})^2 \end{aligned}$$

We apply Ferrari's method to solve the equation. We take the minimum real root as object's landing time and use it to compute g_o. The object will not hit the surface if there is no real root or

$$Angle((x_l - x_{org}, y_l - y_{org}, z_l - z_{org}), (1, 0, 0)) > \alpha \tag{4}$$

To generate G_c, we compute all possible poses of the shield, solve their inverse kinematics and push them into G_c. Specifically, we first find the best shield pose, then we find all shield poses having same position and similar orientation

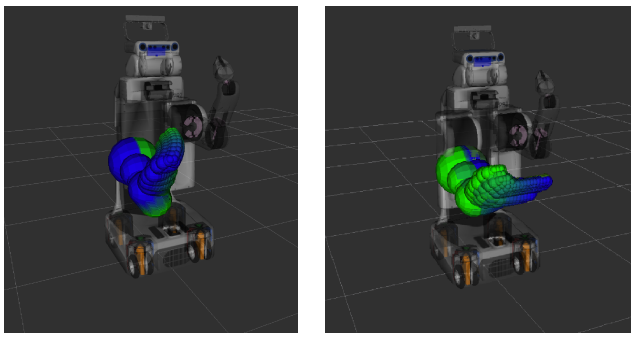
($\delta_{roll} < \epsilon_{roll}, \delta_{yaw} < \epsilon_{yaw}, \delta_{pitch} < \epsilon_{pitch}$) and push their configurations into G_c. The best position is the same with p_l. The best shield orientation is defined to be opposite to v_l, namely the shield surface is perpendicular to v_l.

C. Detail Adjustment

As a whole motion plan is composed of a path from initial state to the attractor and a path from the attractor to goal state, it is a sub-optimal solution. What's more, the longer the second path is, the longer the extra execution time will be. We therefore restrict the radius of subregions with a maximum value r_{max} by adding pseudo code between line 7 and 8 in procedure ComputeReachability

Algorithm 2 Added Part

```
1:  if h(s, s_i) > r_max then
2:      r_i ← r_max
3:      OPEN.push(s)
4:      return (OPEN, r_i)
```



(a) (b)

Fig. 3. When the radius of a subregion is too large, the greedy path (b) may be large compared to the pre-computed path (a), which means the path is far not optimal.

IV. Simulation

TABLE I

List of motion primitives in simulation

x	y	z	yaw	pitch	roll	free angle
± 0.02	0.00	0.00	0	0	0	0.0
0.00	± 0.02	0.00	0	0	0	0.0
0.00	0.00	± 0.02	0	0	0	0.0
0.00	0.00	0.00	± 10	0	0	0.0
0.00	0.00	0.00	0	± 10	0	0.0
0.00	0.00	0.00	0	0	± 10	0.0
0.00	0.00	0.00	0	0	0	± 2.5

We performed simulations on PR2 robot to evaluate our algorithm. The algorithm and simulation environment are implemented using C++ on ROS. An Intel Core i7-7700HQ 2.80GHz CPU machine is employed for simulation. We discretize our graph with a resolution of 2cm in position axes, 10 degrees in Euler axes and 2.5 degrees for the redundant joint. The primitives are defined as subtle movements of the shield in position axes (x, y, z) and orientation axes of Euler angles (roll, pitch, yaw) and subtle movements of the redundant joint (shown in Table I). To be clear, the deviation between poses of the shield under a state and its predecessor/successor is one of the motion primitives. The heuristic function is the Euclidean distance in joint space and the maximum radius of subregion is 0.06. We design the shield to have a radius of 8cm and negligible inertia in Solidworks. For all tests, the surface are defined as :

$$\begin{cases} (x-0.7)^2 + (y-0)^2 + (z-0.9)^2 = 1.56^2 \\ Angle((x-0.7, y-0, z-0.9), (1,0,0)) < 0.4 \end{cases} \quad (5)$$

It's notably that we assume our perception system to be perfect so that we can start planning right after an object shows up. That is, the distance between the robot and object's initial position, which affects accuracy of perception, can be fixed in our simulation. To further reduce the workload, we only plan for the right arm to block the right part of the surface, which obviously will not lead to any impact on results. We evaluate how optimal the overall path is by the ratio between extra path length and total path length. Also, We take number of subregions, planning execution time, query time and search time as other assessment indexes.

We generate 200 random tests for each setting. In each test, an object is initially placed randomly on an arc centered at the PR2 robot with a radius of 8m and a radian of α. Its velocity is randomly sampled within a range. The key idea is to make sure its possible landing poses cover the whole predefined surface. Specifically, we first randomly pick a position on the arc and let the object flies directly to PR2 robot with a random speed in xy plane from 3m/s to 8m/s. Then we randomly generate a pitch angle from -50 degrees to 50 degrees and height from 0.4m to 1.6m for the object to land on. Lastly, the initial height and velocity along z-axis are determined by simulate the movement backward. The preprocessing takes about two hours and there are in total 6534 subregions. We present the results under different initial poses of the shield in Table II. The succesful rate of planning is 100%.

V. Conclusions

In this work, we have presented a preprocessing-based kinodynamic motion planning algorithm that generates collision free trajectories for a manipulator to block flying objects. We build our algorithm on a provable real-time planning algorithm to deal with the high dimensionality and time restraint of our task. We conduct simulation on PR2 robot to show that our algorithm is capable of planning an acceptable path within a short time window. In future work, we will consider uncertainty in the perception system, which involves replanning problem. And we will setup real-world experiment for evaluation.

Acknowledgment

The work is supported by the Search-Based Planning Lab at the Robotics Institute, Carnegie Mellon University and the

Initial pose	Query time[ms]	Search time[ms]	Execution time[ms]	$\frac{\text{Extra path length}}{\text{Path length}}$
$[0.40, 0.00, 0.90, 0.00, 0.00, 0.00]$	$0.82(1.34)$	$12(19)$	$1500(1900)$	0.101
$[0.50, 0.00, 0.90, 0.00, 0.00, 0.00]$	$0.87(1.47)$	$12(21)$	$1200(1800)$	0.124
$[0.60, 0.00, 0.90, 0.00, 0.00, 0.00]$	$0.82(1.52)$	$13(21)$	$1000(1500)$	0.150
$[0.70, 0.00, 0.90, 0.00, 0.00, 0.00]$	$0.79(1.44)$	$13(21)$	$1100(1800)$	0.143

TABLE II

Institute of Artificial Intelligence and Robotics for Society (AIRS), The Chinese University of Hong Kong, Shenzhen. The authors would also like to acknowledge the support of the RISS organizers: Dr. John M. Dolan and Ms. Rachel Burcin.

REFERENCES

[1] G. Walck, R. Haschke, M. Meier, and H. J. Ritter, "Robot self-protection by virtual actuator fatigue: Application to tendon-driven dexterous hands during grasping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2200–2205.

[2] T. Yoshikai, M. Hayashi, A. Kadowaki, T. Goto, and M. Inaba, "Design and development of a humanoid with soft 3d-deformable sensor flesh and automatic recoverable mechanical overload protection mechanism," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4977–4983.

[3] M. Hayashi, R. Ueda, T. Yoshikai, and M. Inaba, "A fall down resistant humanoid robot with soft cover and automatically recoverable mechanical overload protection," in *Advances In Mobile Robotics*. World Scientific, 2008, pp. 1225–1232.

[4] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 816–823.

[5] T. Ishida and R. E. Korf, "Moving target search." in *IJCAI*, vol. 91, 1991, pp. 204–210.

[6] S. Koenig, M. Likhachev, and X. Sun, "Speeding up moving-target search," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007, pp. 1–8.

[7] U. Asif, J. Tang, and S. Harrer, "Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices." in *IJCAI*, 2018, pp. 4875–4882.

[8] F. Islam, O. Salzman, A. Agraval, and M. Likhachev, "Provably constant-time planning and re-planning for real-time grasping objects off a conveyor," *arXiv preprint arXiv:2003.08517*, 2020.

[9] F. Islam, O. Salzman, and M. Likhachev, "Provable indefinite-horizon real-time planning for repetitive tasks," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, no. 1, 2019, pp. 716–724.

Sun Sensor Absolute Heading Determination for Lunar Micro Rover

Thomas Galligani[1], William "Red" Whittaker[2], and Heather Jones[2]

***Abstract*— Planetary rovers estimate their map position by integrating incremental pose using wheel encoding, inertial measurements, and perception of terrain. However, any such dead reckoning method is prone to drift over time; imperfect sensing, calibration and imprecisions in perception and navigation calculations accumulate, creating inaccuracy in the pose estimation that increases with distance traveled. This concern is even greater for smaller rovers such as the MoonRanger rover, which is of particular interest in this research. Smaller rovers are buffeted more, steer with less authority, and cannot incorporate the superior sensors of their larger counterparts. Therefore, a method to accurately sense absolute heading incorporate this into the overall method of position estimation would greatly improve the rover's autonomous relyability. This research formulates and models the means for such sensing of absolute robot heading on the moon by sensing sun angles and utilizing an ephemeris for computing sun-moon orientation variations with time. The technique will vastly reduce directional drift and thereby enable reliable previously unachievable multi-kilometer autonomous lunar rover treks. Since the moon is an airless body, its sun is never diffracted by atmosphere or interrupted by cloud cover - ensuring that observations can be precise. The lack of atmosphere allows for constantly accurate measurements by sun sensors. Fusing absolute pose estimates from sun sensor measurements with inertial measurement unit (IMU) data and other dead reckoning methods provides superior, stable, absolute bearing estimation not possible by any other means. Especially prone to accumulated error is the yaw or heading since errors in the estimate of yaw results in disproportionate drift relative to longitudinal errors. In this paper, we develop a method for pose estimation using sun sensor data for MoonRanger, an autonomous lunar rover that will search for water ice in the south pole of the Moon in 2022. We develop a corresponding simulated sun sensor to aid in validation for MoonRanger's autonomous navigation capabilities.**

***Index Terms*— Autonomous Vehicle Navigation, Space Robotics and Automation, Sensor Fusion**

I. INTRODUCTION

A. Overview

MoonRanger is a small, fast, autonomous lunar rover designed to search for ice in the South Pole region of the Moon. Traditional planetary rovers – controlled by humans throughout the mission – rely on long mission durations (about ten years) to compensate for relatively slow operations. Designed for an eight-day mission, MoonRanger is able to explore much more efficiently by keeping direct human control to a minimum. MoonRanger's small scale, however, exacerbates its slippage and buffeting which incur greater error in achieving its commanded motions than experienced by larger rovers. With these mobility shortfalls MoonRanger's long, autonomous treks without contact with a ground station creates significantly more risk and reliance on the rover's own navigation and control systems. Even more than a traditional planetary rover, it is vital that MoonRanger is able to maintain an accurate estimate of its position and orientation on the lunar surface.

It is common for planetary rovers to rely on a method of dead reckoning for navigation and pose estimation. Dead reckoning is a recursive method of pose estimation, determining the next estimate based on the previous estimate and a measured change. For example, starting from an initial state estimate, accelerometer measurements can be numerically integrated over time and added to the last state estimate. Similar methods using visual and wheel encoder odometry can be used (and combined) to get a more accurate state estimation. However, the primary issue with these dead reckoning methods is drift; due to the additive nature of these methods, errors accumulate, resulting in large error as the distance traveled grows. By fusing one or more dead reckoning methods with an absolute navigation method, this error can be limited substantially.

Unfortunately, many common absolute navigation methods available on Earth are not available to a lunar rover. GPS is unavailable beyond Earth and the lunar magnetic field is not strong and consistent enough to use for navigation [1]. One absolute reference that can be used for pose estimation and navigation is the sun. In fact, using solar navigation is even more suited for the lunar environment than it is

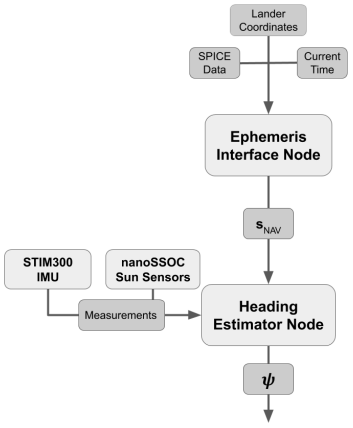
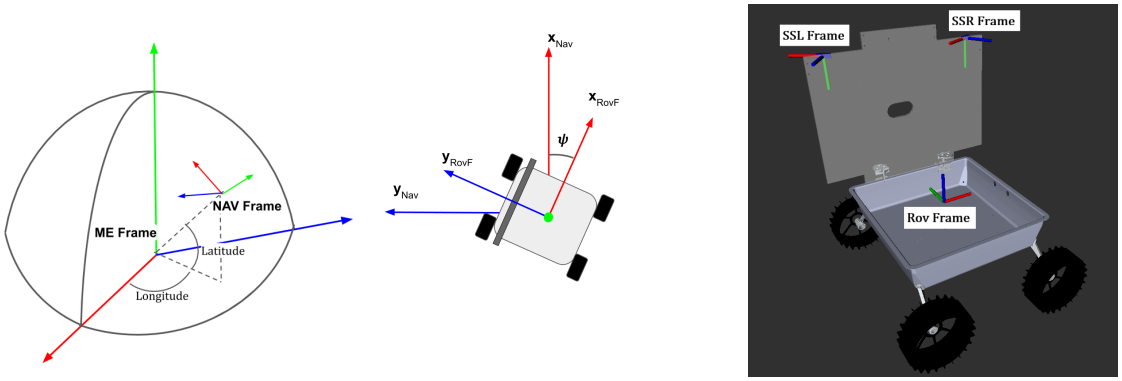


Fig. 1: Basic algorithm flow, using ephemeris and sensor measurements to calculate heading (ψ)

[1]Thomas is a cadet at the Air Force Academy, USAF Academy, CO c21thomas.galligani@afacademy.af.edu

[1]Red and Heather are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA {red, hjones}@andrew.cmu.edu



(a) The NAV frame is a fixed rotation from the ME frame (b) Heading ψ is the difference between the ROVF and NAV frames (c) The SSR and SSL frames are a fixed relative the ROV frame

Fig. 2: Visual definition for various reference frames

on Earth, since the lack of atmosphere permits consistent, reliable measurements without interference from clouds or other atmospheric phenomena. It has been shown that sun sensors can effectively estimate heading for planetary rovers [2] [3] [4]; in this paper, we formulate such an algorithm (visually depicted in Figure 1) for the MoonRanger lunar rover and validate it through simulation.

B. Notation

To describe the pose of the rover, we will rely on rotation matrices, in addition to Euler angles as an intermediary. We will denote a rotation matrix such that $\mathbf{R}_{\text{FR2,FR1}} \in \mathbb{R}^{3,3}$, transforming a vector from the FR1 reference frame to the FR2 reference frame. We use Yaw(ψ), Roll(θ), Pitch(ϕ) to mean the rotation matrices corresponding respectively to a rotation of ψ around the z-axis, θ around the y-axis, and ϕ around the x-axis. We use the notation $\mathbf{s}_{\text{FR}}$ to mean a unit sun vector in the FR frame.

II. PROBLEM SETUP

A. Relevant Mission Parameters

MoonRanger's primary objectives are to explore the region around the lunar South Pole over an eight-day mission to demonstrate autonomous microrover capabilities and to characterize water ice using a Neutron Spectrometer System (NSS). The polar location means that over the mission duration, the elevation angle of the sun will be close to zero (but remain above the horizon) and will change very little [5]. The implications of this is that MoonRanger will rely on almost constant illumination of a vertical solar panel, preferring to travel roughly normal to the sun vector. Therefore, optimizing the sun sensor measurement coverage, two Solar MEMS nanoSSOC-D60 sun sensors located on the top of the solar panel, facing horizontal to the rover body.

B. Reference Frames and Rover Geometry

Here, we define six reference frames. First, the Moon Mean Earth (ME) frame, with an origin at the Moon's center, has a z-axis in line with the lunar rotational pole and an x-axis pointing towards the average "sub-Earth point" - the point on the lunar surface closest to Earth [6]. Next, we define the Navigation (NAV) frame, a topocentric frame which is centered at the lunar coordinates of the landing site with its x-axis pointing due north and its y-axis pointing due west. The transformation from the ME frame to the NAV frame is given by

$$\mathbf{R}_{\text{NAV,ME}} = \text{Yaw}(Lon)\text{Pitch}(Lat - 90^\circ)\text{Roll}(180^\circ) \quad (1)$$

where Lon and Lat are the lunar longitude and latitude of the landing site. Figure 2a provides a visual definition of the relationship between these two frames. Note that we consider only the rotation, not the translation when transforming between frames. Since the distances involved are negligible compared to the radius of the Sun and its distance from the Moon, we can treat light from the Sun as directional across our entire area of interest. Next, the Rover (ROV) frame is centered on the body of the MoonRanger. The y-axis points at the solar panel and the z-axis points up, normal to the surface of the rover body. The Rover Flat (ROVF) frame shares the origin of the ROV frame but shares a z-axis with the NAV frame. The transformation from the ROV to the ROVF frame is given by

$$\mathbf{R}_{\text{ROVF,ROV}} = \text{Yaw}(0)\text{Pitch}(\theta)\text{Roll}(\phi) \quad (2)$$

where θ and ϕ are the pitch and roll measured by on-board inclinometers. The fact that the ROVF and NAV frames share a z-axis means that they differ from each other by a simple rotation about that axis. Figure 2b shows that this offset is the rover's heading. Finally, the Left Sun Sensor and Right Sun Sensor (SSL and SSR) frames are oriented in the with the z-axes pointing 30° right or left of normal to the solar panel surface and a y-axis pointing down. The transformations from the sun sensor frames to the ROV frame are given by

$$\mathbf{R}_{\text{SSR,ROV}} = \text{Roll}(-180^\circ)\text{Pitch}(0)\text{Yaw}(-180^\circ - 30^\circ) \quad (3)$$

$$\mathbf{R}_{\text{SSL,ROV}} = \text{Roll}(-180^\circ)\text{Pitch}(0)\text{Yaw}(-180^\circ + 30^\circ). \quad (4)$$

Figure 2c depicts the relationship between the sun sensor and rover body frames, while Figure 3 shows the overlapping fields of view for the two sensors.

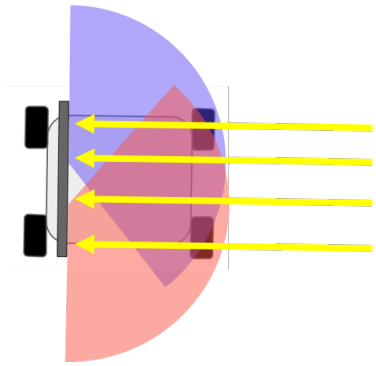


Fig. 3: The field of view for the two sun sensors overlap by 60 degrees in the center of the solar panel

III. HEADING ESTIMATION

A. Heading Determination

To estimate the current heading, we will ultimately compare the sun vector in the ROVF frame, based on the sun sensor and inclinometer measurements, to the sun vector in the NAV frame, calculated from ephemeris data and the current coordinates and time. We begin by computing and normalizing the position vector of the Sun's center relative to the Moon's center in the ME frame; this is our sun unit vector $\mathbf{s}_{\text{ME}}$ in the ME frame. We can then rotate $\mathbf{s}_{\text{ME}}$ into the NAV frame as follows

$$\mathbf{s}_{\text{NAV}} = \mathbf{R}_{\text{NAV,ME}}\mathbf{s}_{\text{ME}}\,. \tag{5}$$

Before we consider the sun sensor measurements, we need to consider three different cases. First, the sun could be out of the field of view of both sun sensors - clearly, in this case, we cannot estimate the heading using the sun sensors. However, this case should be fairly limited for MoonRanger, based on its projected mission design (ie, keeping the sun in the range of the solar panel, and thus sun sensors, will be the preferred mode of travel). Next, the sun could be in the view of only a single sun sensor. In this case, we will estimate the heading with only the measurement from that sensor. Our final case is when the sun is within the FOV of both sensors. In this case, we average the measured sun vectors. For now, we will assume that the sun is in view of only the right sun sensor. Then, we calculate $\mathbf{s}_{\text{SSR}}$ from the measurement taken by the right sun sensor, $\mathbf{m}_{\text{SSR}} = [\alpha, \beta]^T$. We see that based on the geometry in Figure 4 and the restriction $||\mathbf{s}_{\text{SSR}}|| = 1$, we can calculate our sun vector

$$\mathbf{s}_{\text{SSR}} = \begin{bmatrix} \sin(\alpha) \\ \sin(\beta) \\ \sqrt{1 - \sin^2(\alpha) - \sin^2(\beta)} \end{bmatrix}. \tag{6}$$

Next, we construct the transformation from the ROV to the ROVF using the inclinometer measurement $\mathbf{m}_{\text{INC}} = [\phi, \theta]^T$

$$\mathbf{R}_{\text{ROVF,ROV}} = \text{Yaw}(0)\text{Pitch}(\theta)\text{Roll}(\phi) \tag{7}$$

where ϕ and θ are the inclinometer pitch and roll measurements. Now, we can rotate our measured sun vector into the ROVF frame

$$\mathbf{s}_{\text{ROVF}} = \mathbf{R}_{\text{ROVF,ROV}}\mathbf{R}_{\text{ROV,SSR}}\mathbf{s}_{\text{SSR}}\,. \tag{8}$$

Note that we replace $\mathbf{R}_{\text{ROV,SSR}}$ with $\mathbf{R}_{\text{ROV,SSL}}$ if the measurement was taken by the left sun sensor.

Next, we calculate the the azimuth of the sun vector in both the NAV and ROVF frames. We see that

$$\psi_{\text{NAV}} = \text{atan2}(\mathbf{s}_{\text{NAV,x}}, \mathbf{s}_{\text{NAV,y}}) \tag{8}$$

$$\psi_{\text{ROVF}} = \text{atan2}(\mathbf{s}_{\text{ROVF,x}}, \mathbf{s}_{\text{ROVF,y}}) \tag{9}$$

where $\mathbf{s}_{\text{NAV,x}}$ is the x coordinate of the s vector in the NAV frame. Finally, to determine the estimated heading for the rover, ψ, we consider two cases

$$\psi = \begin{cases} \psi_{\text{ROVF}} - \psi_{\text{NAV}} & \text{if } \psi_{\text{ROVF}} < 0 \\ \psi_{\text{NAV}} - \psi_{\text{ROVF}} & \text{if } \psi_{\text{ROVF}} \geq 0 \end{cases}. \tag{10}$$

B. Implementation

We implemented this algorithm in the C++ version of the Robot Operating System (ROS). The primary functionality of this algorithm is contained in two nodes: one dealt with ephemeris data and determined the sun vector in the NAV frame and one which used this data along with the sensor measurements to determine the heading. The ephemeris node uses JPL's CSPICE toolkit, a C implementation of the SPICE observation geometry system. This toolkit provides a method of interfacing with precise navigational data compiled by NASA for planetary exploration missions. We use this ephemeris data to determine the location of the Sun relative to the Moon. Our second node performs the comparison from this known relative positional data and sensor observations to estimate MoonRanger's heading.

IV. VALIDATION

To validate our method, we turn to simulation. The sun sensors which will fly on MoonRanger are high precision, high cost, and high lead time - their use is unnecessary for the proposes of this work. However, the previous simulation used for testing the software for autonomous navigation included most relevant sensors (stereo cameras, IMU, wheel

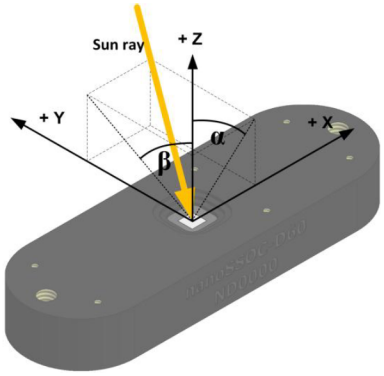


Fig. 4: Measurement angles are based on the angular distance of the sun vector from the z-axis [7]

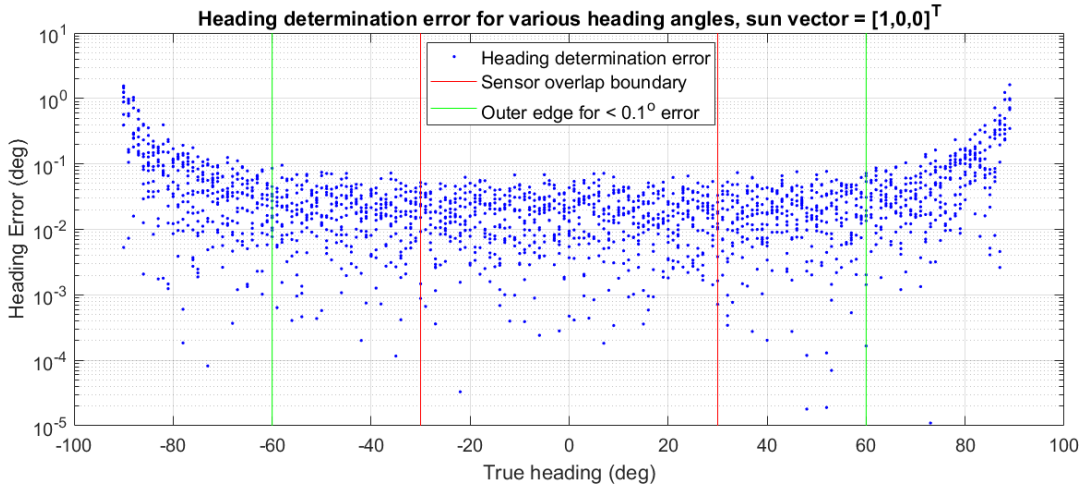


Fig. 5: Error increases sharply at the edges of sensor range, but is controlled within the region where both sensors overlap

encoders, etc) but was lacking simulated sun sensors. The Gazebo robotics simulator - used for this purpose - lacks an easily applied simulated sun sensor, unlike many other, more common sensors. To fix this, we developed a simulated sun sensor in ROS, using SPICE ephemeris data and a knowledge of the simulated rover's true orientation relative to the global frame of the simulation.

A. Simulated Sensors

First, the ROS sensor receives $\mathbf{s}_{\text{NAV}}$ for the particular landing site and time from the ephemeris interface node and rotates it into the ROV frame using the known simulated rover orientation. Next, we can rotate this vector into the SSL and SSR frames (both known, static transformations) and calculate the nominal sun sensor measurements for the left and right sensors. We simply compute

$$\overline{\mathbf{m}}_n = \begin{bmatrix} \arcsin(\mathrm{s}_{\mathrm{ssn},y}) \\ \arcsin(\mathrm{s}_{\mathrm{ssn},y}) \end{bmatrix} \tag{11}$$

where $\mathrm{s}_{\mathrm{ssn,y}}$ and $\mathrm{s}_{\mathrm{ssn,y}}$ are the x and y components of our sun vector in the n sun sensor frame and $\overline{\mathbf{m}}_n$ is our nominal measurement. We can then add some Gaussian white noise, with the precision of MoonRanger's flight sun sensors. Then our simulated measurements become

$$\mathbf{m}_n = \overline{\mathbf{m}}_n + \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} \tag{12}$$

where $v_\alpha, v_\beta \sim \mathcal{N}(0, \frac{1}{6})$. Note that we measure angles in degrees. The last step is to simulate sun sensor error codes. The nanoSSOC gives a code 0 for a normal, successful observation. The sensor gives a code 11 if the measured light intensity is less 80 percent of the expected value of 1366 $\frac{\mathrm{W}}{\mathrm{m}^2}$ (ie, the light being measured is reflected or from a different source and cannot be assumed to be a correct sun vector. To simulate this situation, we set our error code to 11 when the sun vector is out of the view of the sensor - when either simulated measurement angle has a magnitude greater than 60^o [7]. Therefore, our simulated sun sensor node publishes six values: two simulated measurement angles and one error code for each sensor.

B. Testing

With our simulated sensor, we can test the heading determination method over a representative domain of rover orientations. We step over the ephemeris calculation and choose $\mathbf{s}_{\text{NAV}} = [1, 0, 0]^T$ (ie, the sun points due West). This means that when rover heading is zero, the sunlight is normal to the solar panel (the same situation depicted earlier in Figure 3). We simulate sensor readings with heading ranging from -90^o to 90^o (the combined range of the sensors) and roll ranging from -20^o to 20^o.

C. Results

Figure 5 shows the heading determination error for various true rover headings. We have divided this plot into 3 regions, based on the upper bound of the error they experienced. In the center, bounded by vertical red lines, is the region where the field of view of the two sensors overlap. Here, the error remains controlled, unlike other regions. Beyond the overlapping region, the error begins to grow slightly - the green vertical lines mark the point where (on both the positve and negative sides) the error reaches 0.1 degrees. Beyond this boundary - where heading is $\pm 60°$ - error increases sharply - maxing out at over 1 degree for the outer limit of the sensors' view. This implies that the overlap of the sensors in the center region effectively controls the error which is inherent to the edge of a particular sensor's view. While the error in the outer region increases to unacceptable levels to be trusted by the overall pose estimation method, knowledge of this increased error at the edges can inform the way this method interfaces with the overall position estimation algorithm. Giving less weight these edge case measurements, we can benefit from the method when it performs well (in the central regions) and avoid projecting its limitations (at the edges) to the overall pose estimation method.

V. FURTHER WORK

To further validate this method, it would be valuable to conduct testing with real, physical sun sensors. Additionally, an important next step for this work is integrating the solar heading determination algorithm to the greater pose estimation EKF; this involves mathematically formulating the heading determination error to fuse it with the wheel encoder and accelerometer measurements. With these two methods integrated together, MoonRanger's pose estimation method will be significantly more stable and reliable over long autonomous treks.

ACKNOWLEDGMENT

The author appreciates the Carnegie Mellon MoonRanger team for providing the context and collaboration that guided this research. The author thanks the United States Air Force Academy for the opportunity and support to pursue this research and contribute this technology with broad impact and enabling relevance to an upcoming lunar exploration.

REFERENCES

[1] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Rover navigation using stereo ego-motion," *Robotics and Autonomous Systems*, vol. 43, p. 215–229, 2003.

[2] M. Ilyas, K. Cho, S. Park, , and S.-H. Baeg, "Absolute navigation information estimation for micro planetary rovers," *International Journal of Advanced Robotic Systems*, vol. 13:42, 2016.

[3] A. Lambert, P. Furgale, T. D. Barfoot, and J. Enright, "Visual odometry aided by a sun sensor and inclinometer," *IEEE Transactions on Automatic Control*, 2011.

[4] T. Barfoot, J. R. Forbes, and P. T. Furgale, "Pose estimation using linearized rotations and quaternion algebra," *Acta Astronautica*, vol. 68, pp. 101–112, 2011.

[5] D. B. J. Bussey, P. D. Spudis, and M. S. Robinson, "Illumination conditions at the lunar south pole," *Geophysical Research Letters*, vol. 26, pp. 1187–1190, May 1999.

[6] "A standardized lunar coordinate system for the lunar reconnaissance orbiter," LRO Project White Paper, May 2008.

[7] "nanossoc-d60 technical specification, interfaces operation," Technical Specifications, May 2016.

Using Process Mining to Analyze Children's Interactions with RoboTutor

Tianjian Huang[1] and Jack Mostow[2]

***Abstract*— When using an Intelligent Tutoring System (ITS), ideally students are engaged while interacting with it. From the developer's perspective, detecting how engaged or disengaged students are when interacting with ITS provides clues to improve the software. This paper presents an approach to analyze student-tutor interaction behaviors by applying Educational Process Mining (EPM) techniques to log data from RoboTutor, a Swahili literacy and numeracy tablet tutor for children. The results show that process mining (PM) methods can extract information from the log data to help in understanding student-tutor interaction patterns and discovering potential improvements.**

***Index Terms*— Process Mining, Educational Data Mining, Intelligent Tutoring System, Disco**

I. INTRODUCTION AND RELATION TO PRIOR WORK

An ITS is an adaptive system that provides a learning environment to students, monitors and analyzes their actions to detect if they are learning the knowledge being taught, and provides feedback [1]. One focus in the field of ITS is engagement. The definition of engagement remains elusive, or at least it is a broad and complex concept [2]. To simplify our study, we only consider behavioral and cognitive components of engagement proposed by Fredricks, Blumenfeld, and Paris, which encompass students' participation and performance in learning [3]. Engagement is a generally acknowledged critical factor in learning. Studies show that engagement is positively correlated with learning, while disengagement is negatively correlated with learning and will significantly reduce learning outcomes [4], [5]. The importance of engagement motivates scientists to detect engagement or disengagement behaviors. Prior studies used facial expressions, posture, response time, task performance and problem-specific features as well as mouse movements for automatic detection of disengagement behaviors in ITS such as off-task behavior and gaming [6]–[10]. Particularly, [8], [9] and [10] applied machine learning methods to ITS log data and obtained promising results. However, those approaches have non-negligible limitations; they depend on various hardware sensors (e.g. camera, Body Pressure Measurement System [7]), and require accurately labeled training data.

As a new technology emerging from Educational Data Mining (EDM), EPM analyzes log data gathered from educational environments and provides insights into educational processes by discovering sequential patterns in them [11]. This paper illustrates an EPM approach to analyzing student behaviors that underlie RoboTutor log data. We first extract behavioral and performance information from the log data. Based on the extracted information, we create an abstract model of the underlying real-life process. Using this process model, we infer and predict engagement and disengagement. In contrast to the approaches mentioned before, EPM provides a data-driven way to reveal behavioral patterns underlying ITS log data without hardware sensors or manually labeled training data.

The rest of the paper is organized as follows: Section 2 describes the RoboTutor log data we analyzed. Section 3 specifies our EPM methodology for analyzing student-tutor interactions. Section 4 reports our results. Finally, section 5 summarizes contributions, limitations, and future work.

II. DATA SET

The present study used RoboTutor log data collected by Global Learning XPRIZE field staff from 28 villages in Tanzania between 01/18/2019 and 03/01/2019. RoboTutor is an Android tablet tutoring app in the Global Learning XPRIZE competition [12]. A RoboTutor *session* starts when RoboTutor launches and ends when it exits (or crashes). RoboTutor contains thousands of *tutors* to teach various literacy and numeracy skills. A tutor *instance* ("instance" for short) spans the time interval from starting a tutor to either completing it or backing out without completing it (or crashing). Thus, a session contains zero or more instances. For example, Bubble Pop tutors teach children to recognize numbers, letters or words. Fig. 1 is a screenshot of a Bubble Pop tutor call *Bpop.wrd*. The tutoring problem consists of an audio prompt and three "bubbles" representing three choices. To give an answer children need to tap on one of the bubbles, and the correct answer is the bubble corresponding to the audio prompt.

RoboTutor creates different types of log data during each session, namely PERF logs, VERBOSE logs, and CRASH logs. A PERF or VERBOSE log is a JSON file containing a sequence of JSON objects, each of which records an action by a student or the RoboTutor. As Fig. 2 shows, a JSON object is a sequence {...} of one or more key-value pairs. The value can be a literal "...", a sequence [...], or another JSON object (e.g. the value of the key "data"). PERF logs and VERBOSE logs are similar in structure but contain different kinds of records. PERF logs just record user actions, such as switching to another tutor or solving a problem. VERBOSE

[1]Tianjian Huang is a senior undergraduate in the School of Data Science, Chinese University of Hong Kong, Shenzhen, China. 117010099@link.cuhk.edu.cn

[2]Jack Mostow is with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213. mostow@cs.cmu.edu

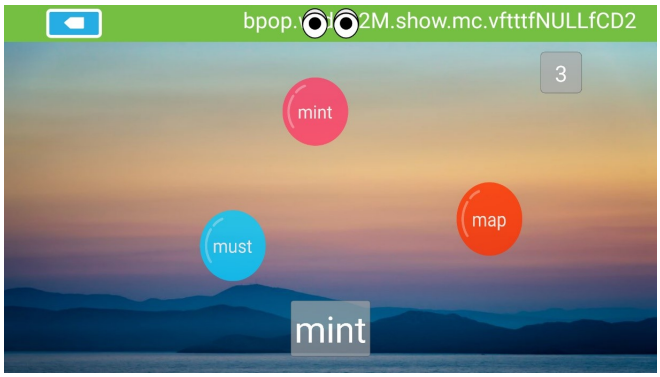


Fig. 1. A screenshot of Bpop.wrd tutor (on an undersized screen that distorts the layout). This tutor teaches children to recognize words.

logs additionally record internal RoboTutor actions invisible to users, such as switching process context from one tutor to another, or calling Java code to display a new problem. As a result, the VERBOSE logs are about 5 times as long as PERF logs. Finally, CRASH logs are generated only when RoboTutor crashes, and contain the error message and stack trace to aid in debugging.


```
{
"type": "LOG_DATA",
    "tutor": "bpop.wrd:m2M.show.mc",
    "class": "INFO",
    "tag": "RTag",
    "time": "1596523024338",
    "data": {
        "target": "node.root",
        "name": "INTRO_STATE",
        "start State": "null",
        "mapType": "moduleMap",
        "mapName": "INTROSTATE"
    }
}
```


Fig. 2. An example of a JSON object in Bpop.wrd VERBOSE log.

To analyze student-tutor interactions, we need records of both user and RoboTutor actions. Therefore, we chose to analyze VERBOSE logs rather than PERF logs. Given the amount of data (12.7 GB, with over 20,000 files), we selected a random sample of all VERBOSE log files, thereby obtaining 5,000 files to analyze. RoboTutor has different types of tutors. To simplify our study, we focused on just three of the most common ones:

- **Bubble Pop** is illustrated in Fig. 1.
- **Akira** is a time-pressured multiple choice activity in the form of a race car game.
- **Story** tutors display picture stories for RoboTutor or the child to read aloud.

III. Approach and Analysis

Our approach has four stages:

1) Specify a research goal.
2) Extract data from RoboTutor VERBOSE logs into an event log.
3) Create a control flow model of the process from the event log.
4) Use the model to analyze the data from the process.

We now describe each stage in more detail.

A. Specify a Research Goal

We started with planning research goals and questions, a prerequisite for applying any process mining methods [13]. To discover how EPM can analyze RoboTutor log data and infer engagement or disengagement, we used the following criteria as our definition of engagement and disengagement behaviors:

Engagement:

1. The tutoring instance is completed, as signified by a program-defined, tutor-specific normal final step. For example, a Bubble Pop tutoring instance normally ends with the FINAL_SCORE step, which displays the final score.

Disengagement:

1. Incompletion of a tutoring instance, signified by a BACKBUTTON ending step triggered when the child taps the "Back" arrow icon at the top left of the screen, which terminates the instance and returns to RoboTutor's menu for selecting a tutor.
2. Taking too long according to a heuristic criterion, namely lasting longer than 95% of instances of the same tutor type in the sample, whether due to prolonged inactivity or excessive random guessing.

After operationalizing engagement and disengagement, we formulated questions about three aspects of tutor instances:

1. **Completion/incompletion status:**
   How many completions and incompletions does each tutor have? How many incompletions end with BACKBUTTON? How many incompletions have other ending steps?
2. **Duration of tutoring instances:**
   For each tutor, what is the distribution of instance durations? Does it differ for completion vs. incompletion?
3. **Path patterns:**
   What is the most frequent sequence of steps in each tutor? What is its most frequent ending step? Does it have any other ending steps?

B. Extract Data from RoboTutor VERBOSE Logs into an Event Log

This stage aims to convert nested JSON log files into a uniform event log. In the terminology of PM, an *event log* corresponds to a single *process*, and each event refers to a single *process instance* call *case*, where each *case* consists of one or more *activities* [13]. In a PM perspective, a specific type of tutor (e.g. Bpop.wrd which teaches word recognition) is considered a single process. Fig. 3 illustrates the control flow of all Bubble Pop tutors as a directed graph.

Each node represents a *basic action*, either a user action (e.g. PLAY_CORRECT or PLAY_WRONG) or a RoboTutor action (e.g. UPDATE_STIMULUS, which sets the next problem). Each tutor instance corresponds to one event in the event log. Each basic action constitutes an event log activity.

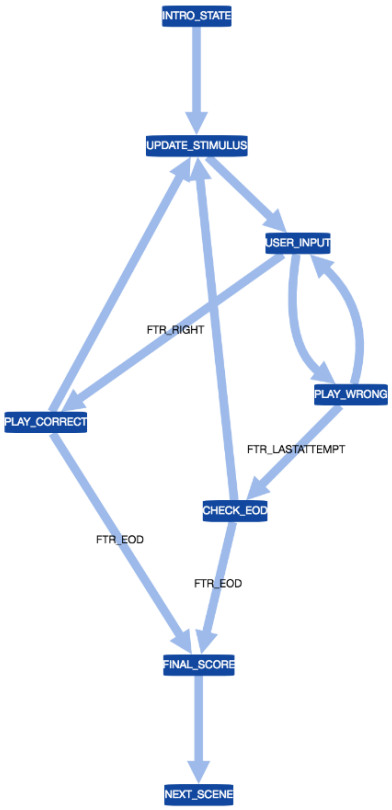


Fig. 3. Control flow design of Bubble Pop tutors.

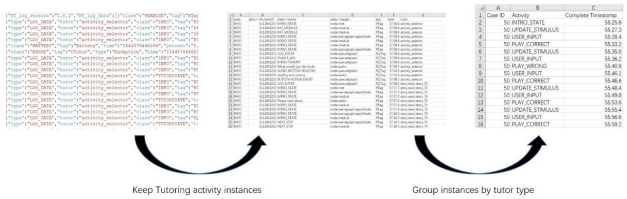


Fig. 4. Illustration of JSON-CSV conversion.

The PM tool cannot directly read JSON, so we wrote a Python 3.8 script to convert JSON files into CSV files, extract data necessary for analysis, and format it to be readable by PM tools. RoboTutor logged millions of actions in VERBOSE log files. As Fig. 4 shows, the conversion script kept only records of tutoring instances, filtering out records of backend actions such as context switching. Then we grouped cases by tutor type, and stored cases of each type into a separate CSV file for each type (e.g. bpop.wrd.csv, story.read.csv, etc.). We thereby obtained 96 MB of CSV event log files.

C. Create a Control Flow Model of the Process from the Event Log

To describe the process underlying an event log, we used the PM tool Disco to create a control flow model of it. Disco is a general purpose commercial process mining tool that has been used in many studies [11]. Fig. 5 and Fig. 6 display Disco's GUI importing an event log for Bpop.wrd and converting it into a process model. Basically, a process model is a control flow diagram inferred from the event log that describes the activity sequence pattern of the process.

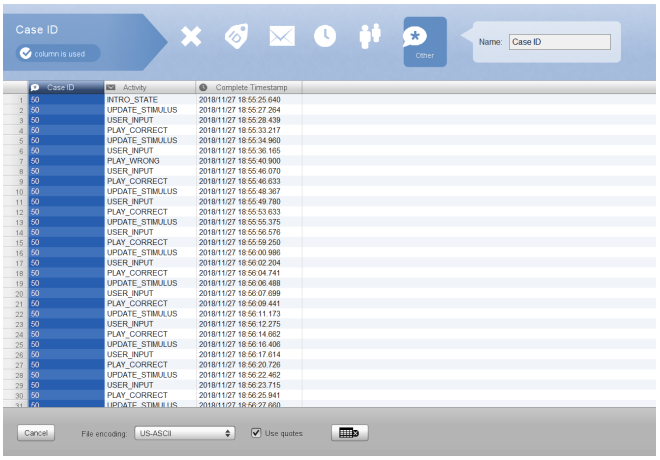


Fig. 5. Example of event log imported from Bpop.wrd.CSV.

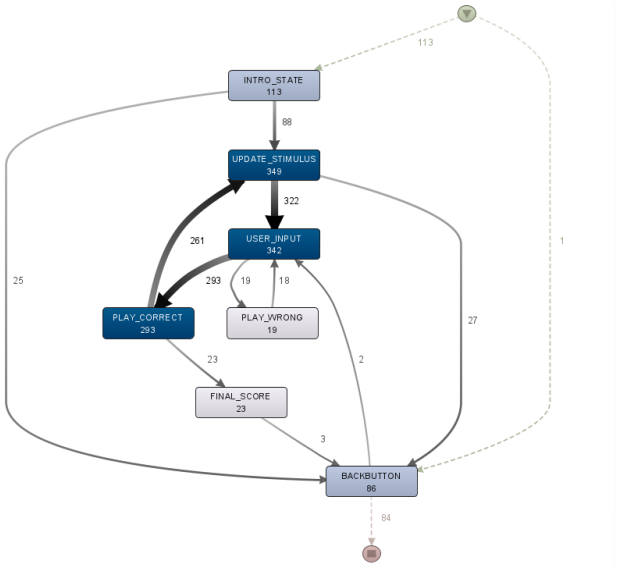


Fig. 6. Example of process model mined from Bpop.wrd event log.

D. Use the Model to Analyze the Data from the Process

Disco can enhance the process model by integrating it with event log statistics such as case duration and activity frequency. We used three filters provided by Disco to generate integrated process models:

1. **Variation filters** group cases according to path pattern, treating cases with the same activity sequence as one variation.
2. **Attribute filters** select cases with or without certain activities, e.g. all cases that contain scoring activities.
3. **Endpoint filters** select cases according to their starting or ending activities, e.g. all cases ending with BACK-BUTTON.

The integrated process model enabled us to analyze student-tutor interaction behavior in RoboTutor and answer the research questions, as we now describe.

IV. Results

A. Completion/Incompletion Analysis

The endpoint filter of Disco can disaggregate the event log by ending activities, separating cases in each event log into three completion status categories: complete with the normal ending activity (defined by RoboTutor developers), incomplete ending with BACKBUTTON, and incomplete with other endings. Fig. 7 and Table I display statistics for six types of Bubble Pop tutors and four types of story tutors.

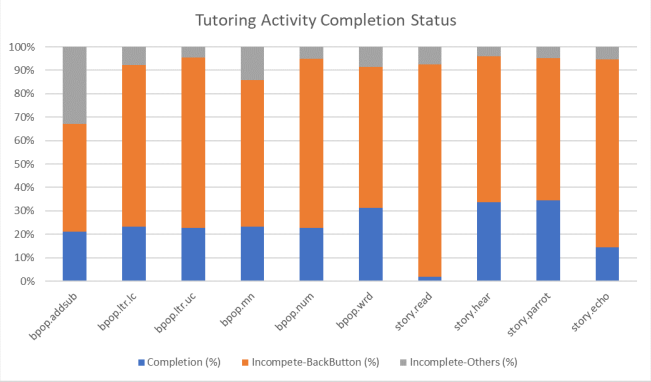


Fig. 7. Completion/incompletion rate of different tutors.

a) **Completion rate:**
The average completion rate is 24.3% for Bubble Pop tutors, versus only 21.7% for Story tutors. We infer that children are likelier to complete Bpop tutors than story tutors.

b) **Incompletion rate:**
Average total incompletion rates are nearly three times higher than completion rates, which indicates that most instances of these tutors were not completed. Of the incompletions, over 80% ended with BACKBUTTON, i.e. most of them occurred because children actively backed out of the tutor.

c) **Outliers:**
From Fig. 7 we observed that Bpop.addsub has a noticeably higher ratio of incompletion with other ending activities. Such endings are not normal, but occur when RoboTutor crashes or the tablet battery runs out due to prolonged use without recharging. The maximum case duration of analyzed Bpop.addsub instances was 7 mins 52 secs, which is not long enough for the battery to run out. Therefore, we infer that the Bpop.addsub tutor suffers from a high crash rate likely caused by a software bug.

Fig. 7 also shows that Story.read tutors have a high incidence of incompletion via BACKBUTTON, with 90% of children choosing to quit and only 1.8% completing the story. Path pattern analysis in part D of this section also indicates that backing out of story tutors occurred mostly at the beginning of the story. Story.read tutors prompt the child to read aloud. The difficulty of this task for young or struggling readers is compounded by RoboTutor's high rejection rate of words read aloud, which evidently frustrated children so much that they quickly learned to back out of such tutors. In contrast, incompletion was much lower for story.hear tutors, which read aloud to the child, and story.parrot tutors, which prompt the child to "repeat after me." Incompletion was somewhere in between for story.echo tutors, which prompt the child to read aloud, but respond after each sentence by reading it fluently to the child.

B. Case Duration Analysis

Fig. 8 illustrates the case duration distribution of four tutors Bpop.num, Bpop.wrd, Akira and Story.read. We observed that:

a) For completed cases, distributions of case durations have clear concentrations, but their location differs by tutor. Case durations peak around 1.5 minutes for Bpop.num, Bpop.wrd, and Akira, and at around 5 minutes and 20 minutes for Story.read.

b) For incomplete cases, case duration distributions are very similar. There are two distinctive features. First, most of the cases ended almost immediately, since there is a high peak at around 0 minutes. Specifically, most of the cases ended within a few seconds. Second, a few extreme outliers have very long durations, since there is a long tail with almost no cases.

C. Path Pattern

Disco can organize cases according to the path pattern of a process. Fig. 9 shows an example of applying the variation filter to identify the most frequent path in Bpop.wrd, shown on the right.

a) For completed cases of Bpop.num, Bpop.wrd and Akira, we discovered that "straight corrects" (no wrong answer during the entire instance) are the most frequent path, accounting for 53.7%, 41.4%, and 35.5% of cases, respectively. However, straight corrects in Story.read account for only 5% of all completed cases. The low straight correct rate in Story.read may indicate a high rate of false rejections.

b) For incomplete cases ending with BACKBUTTON, we observed that the most frequent path ends before practice begins: 71.2% for Bpop.num, 55.6% for Bpop.wrd, 60.1% for Akira, and 65.6% for Story.read, suggesting

TABLE I

WEIGHTED AVERAGE COMPLETION/INCOMPLETION RATIO OF BUBBLE POP AND STORY TUTORS.

Tutor	Completion (%)	Incomplete-Total (%)	Incomplete-BACKBUTTON (%)	Incomplete-Others (%)
Bpop	24.3	75.7	61.8	13.9
Story	21.7	78.3	72.9	5.4

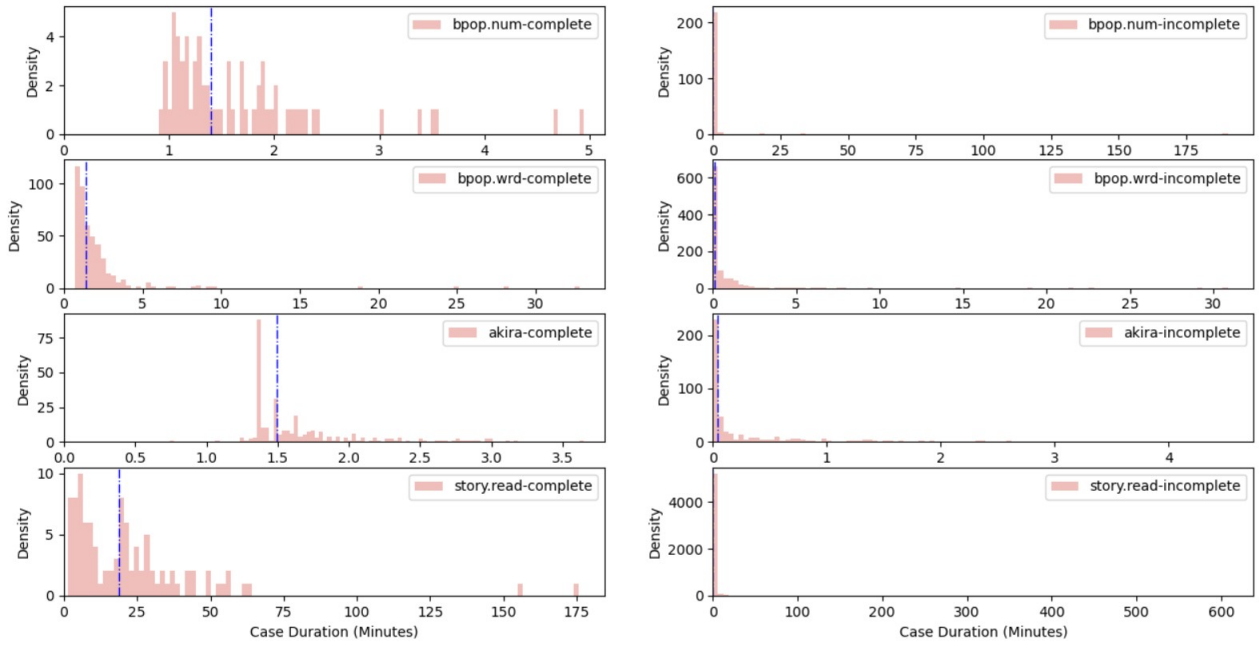


Fig. 8. Case duration distribution of four tutors, with completed cases on the left and incomplete cases on the right. Blue vertical lines show the median of each distribution. Notice that the medians for incomplete cases are very close to zero.

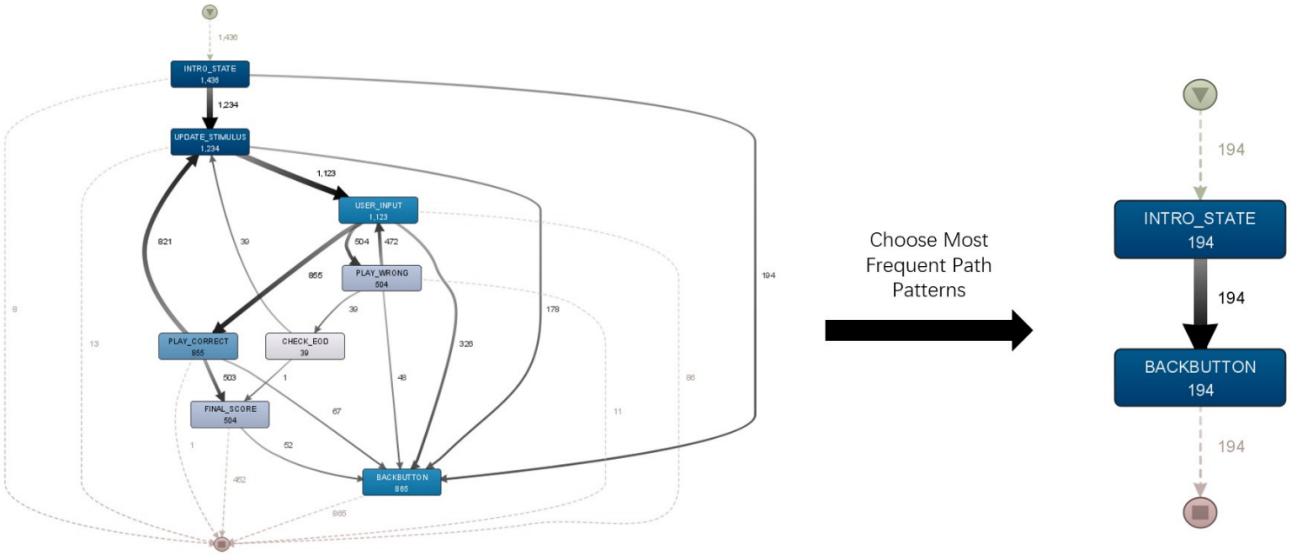


Fig. 9. An example application of variation filter on Bpop.wrd event log. The process model on the left corresponds to the original event log. The process model on the right is generated by selecting the most frequently occurring activity sequence in the original event log.

that children back out as soon as they recognize the tutor.

c) Frequent ending paths for incomplete cases ending with other activities may indicate points where crashes tended to occur. For Bpop.num, 13 out of 15 cases end after USER_INPUT, and only 2 out of 15 cases end after PLAY_CORRECT. For Bpop.wrd, 90 out of 125 cases end after USER_INPUT. This implies that crashes happened during or after USER_INPUT. For Story.read, the session ends in 314 out of 417 cases after LISTEN, implying that crashes tended to occur during or after this activity.

V. Conclusion

Contributions: We presented an EPM method for analyzing RoboTutor log data. We showed that PM can extract key information and help in understanding student-tutor interaction behaviors and discovering potential aspects to be improved. Based on our analysis results, we can infer three recommendations for RoboTutor:

1. Bpop.addsub may suffer from high crash probability, and developers need to test this tutor again.
2. Story.read may suffer from false rejection issues. Developers should tune the speech recognizer.
3. Information from part 1, 3, and 4 indicates that the incompletion rate is 3 times higher than the completion rate on average, and most of the incomplete cases end with BACKBUTTON. RoboTutor may need Bubble Pop and story tutors redesigned so that children are more inclined to complete them.

Limitations and future work: Our analysis used Disco as our main tool for EPM. As a general purpose commercial PM tool, Disco provides basic PM functions such as discovering process models from historical event logs, but does not perform all state-of-the-art PM techniques [11]. Further analysis (e.g. dynamic data analysis of a running tutor) could be done via ProM, a generic open-source PM tool which is the most complete in functionality and most commonly used [11], [14].

As our analysis displayed, EPM can obtain various results, but their value depends on how they are interpreted. Unfortunately, there is a lack of research on a universal approach of interpreting results [14]. The lack of a portable solution (i.e. a common methodology to apply to a wide range of real world processes) makes EPM a case-by-case study and hard to implement as a repeatable service [14]. Future research on EPM can focus on finding an all-encompassing framework which can handle the most general needs.

ACKNOWLEDGMENT

This work would not have been possible without the support from the Fluxicon team, especially Dr. Anne Rozinat and Dr. Christian W. Günther for generously providing an academic license to use Disco. The author would also like to thank Dr. Jack Mostow, Rachel Burcin, Dr. John Dolan and RoboTutor team member Nirmal Patel for their tireless support throughout the summer. The first author was a Robotics Institute Summer Scholar supported by the Institute of Artificial Intelligence and Robotics for Society (AIRS), Chinese University of Hong Kong, Shenzhen.

References

[1] F. Amastini, "Intelligent tutoring system," *Ultima InfoSys: Jurnal Ilmu Sistem Informasi*, vol. 5, no. 1, pp. 1–7, 2014.

[2] S. D'Mello, E. Dieterle, and A. Duckworth, "Advanced, analytic, automated (aaa) measurement of engagement during learning," *Educational psychologist*, vol. 52, no. 2, pp. 104–123, 2017.

[3] J. A. Fredricks, P. C. Blumenfeld, and A. H. Paris, "School engagement: Potential of the concept, state of the evidence," *Review of educational research*, vol. 74, no. 1, pp. 59–109, 2004.

[4] S. Craig, A. Graesser, J. Sullins, and B. Gholson, "Affect and learning: An exploratory look into the role of affect in learning with autotutor," *Journal of Educational Media*, vol. 29, 11 2004.

[5] R. S. Baker, A. T. Corbett, K. R. Koedinger, and A. Z. Wagner, "Off-task behavior in the cognitive tutor classroom: when students" game the system"," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004, pp. 383–390.

[6] R. Sawyer, A. Smith, J. Rowe, R. Azevedo, and J. Lester, "Enhancing student models in game-based learning with facial expression recognition," in *Proceedings of the 25th conference on user modeling, adaptation and personalization*, 2017, pp. 192–201.

[7] S. S. D'Mello, P. Chipman, and A. Graesser, "Posture as a predictor of learner's affective engagement," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, no. 29, 2007.

[8] R. S. Baker, "Modeling and understanding students' off-task behavior in intelligent tutoring systems," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007, pp. 1059–1068.

[9] J. E. Beck, "Using response times to model student disengagement," in *Proceedings of the ITS2004 Workshop on Social and Emotional Intelligence in Learning Environments*, vol. 20, 2004.

[10] S. Cetintas, L. Si, Y. P. P. Xin, and C. Hord, "Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques," *IEEE Transactions on Learning Technologies*, vol. 3, no. 3, pp. 228–236, 2009.

[11] A. Bogarín, R. Cerezo, and C. Romero, "A survey on educational process mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 1, p. e1230, 2018.

[12] J. Mostow, *RoboTutor - RoboTutor - Carnegie Mellon University*, 2020 (accessed August 2, 2020). [Online]. Available: https://www.cmu.edu/scs/robotutor/

[13] W. Van Der Aalst, "Data science in action," in *Process mining*. Springer, 2016.

[14] M. A. Ghazal, O. Ibrahim, and M. A. Salama, "Educational process mining: a systematic literature review," in *2017 European Conference on Electrical Engineering and Computer Science (EECS)*. IEEE, 2017, pp. 198–203.

A Machine Theory of Mind Approach to Agent Intervention

Seth Karten[1], Dana Hughes[2], and Katia Sycara[3]

***Abstract*— Software agents are used in a variety of scenarios from shopping bots to emergency response operators. While most agents are used to provide a service for humans, they are also used to guide the actions of humans. In order to provide informative aid to humans, agents must be able to model a human's mental state, which includes their desires, beliefs, and intentions [1]. Existing methods that attempt to model humans make use of algorithms that do not generalize well to multiple policies. It needs to be taken into account that humans may change their policies based on new information. By modeling the theory of mind, agents can learn the policies of other agents, even human agents. However, current implementations are limited when it comes to recognizing false beliefs. In this work, we determine scenarios in a search-and-rescue problem, simulated in Minecraft, that require interventions; that is, they require a change in policy for agents to optimally perform. Using a theory of mind model, our work evaluates human data to track internal states and first-order false beliefs, and then use the information to identify intervention scenarios. As a parallel approach, we use curriculum-based deep reinforcement learning to train an expert agent on the task. The agent's learned value function is used to determine when an intervention is necessary; that is, the value function acts as a second-order belief predictor of the human player's action-policy.**

I. INTRODUCTION

Using software agents to help guide human policies is beneficial in situations where the human has forgotten information required to complete a task or when a human chooses an action that does not reflect their current policy. One may use an agent to make interventions, or, more broadly, to assist users in a variety of actions, such as emergency response operators and search-and-rescue operations. An intervention is an instruction to a human player that helps them correct their behavior when they are performing a task sub-optimally. When guiding humans, it is challenging to understand the reasons behind their actions. However, understanding a human's intentions, allows one to better guide a human's actions. Thus, it is important for agents to model a human's internal mental state in order to accurately guide a human during a task.

Inverse planning has been used to create generative models of human action [2]. These models are able to show how agents can infer the behavior of others given limited observations. Models have been shown to learn the behavior of other agents by anticipating the opposing agents' policies as an update parameter, creating a model-free learning problem [3].

[1]Seth Karten is with the Department of Computer Science, Rutgers University, New Brunswick, NJ 08854, USA seth.karten@rutgers.edu

[2,3]Dana Hughes and Katia Sycara are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {danahugh, sycara}@andrew.cmu.edu

Fig. 1: The player is searching through the office building that has undergone a catastrophe, creating a need for victims to be rescued. Above is a view of a human player triaging a high-risk victim (yellow) in the Minecraft simulated environment. There is a low-risk victim (green) in the background. The player is also holding a torch to light one's surroundings due to a loss of lighting. The time remaining before all the victims succumb to their injuries is on the middle-right.

Generative modes split an uncertain task into an integrated planning and execution problem [4]. However, generative models do not adequately represent a human's state of mind in the approach or solution to the task. That is, generative models do not generalize well to a variety of policies at once. While these models intend to learn a human model, they focus their efforts towards static artificial policies. The models need to be retrained to account for different agent policies or even normal shifts in a human agent's policy.

More recent work investigates various artificial intelligence methods which attempt to recognize the intent of humans [5]. With this information, the agent must determine what information is important and when is it necessary to provide the human this information. The methods require solving a new Markov Decision Process each time a new planning behavior becomes the norm for the human [5]. The lack of generalization makes these methods intractable.

State-of-the-art work creates a theory of mind framework [6], which represents the mental state of humans. The theory of mind framework creates a knowledge base that can map observations to a latent representation, which can then be used to determine actions and beliefs. The framework is able to generalize an agent's behavior based on a smaller number

of observations than it was originally trained [6]. In this model, the agent uses meta-learning to observe the behavior of new agents and develop information-rich priors. While the theory of mind model is able to learn random, algorithmic, and deep reinforcement learning agents in a variety of tasks, and has been able to learn agent behaviors and recognize false beliefs, these predictions have not been used to guide agents.

Related to the theory of mind framework, inverse reinforcement learning has used to train new agents to model another observed agent. In inverse reinforcement learning, the method extracts the reward function from data that shows the observed agent's policy. The reward function is then used to train a new agent to mimic the observed agent [7]. However, this method is data heavy, and thus was not be pursued in this work.

It has not been studied how using a theory of mind framework can influence agent behavior through interventions. One similar work evaluated whether theory of mind neural models are able to recognize false beliefs in a question answering model. False Beliefs can be first-order, which occur when a player has a different belief about their environment than what is true, or second-order, when an agent has a different belief about the player's internal states than what is true [8]. However, the models are unable to track the inconsistencies in the data.

Curriculum-based reinforcement learning has been used to train agents to solve complex challenges with fewer iterations. Curriculum-based reinforcement learning also trains agents on a series of more complex challenges in the hopes that the agent is able to learn the new task with the previous learned experience [9]. In this work, we collect and analyze human trials of a search-and-rescue simulation setup in a Minecraft-based environment. Using this data, we are able to determine the strategy behind the human players' actions. We identify human first-order false beliefs and use them to analyze intervention scenarios that can be classified from human data. As a parallel approach, we use curriculum-based reinforcement learning to train an expert agent on the task. The agent's learned value function is used to determine when an intervention is necessary; that is, the value function acts as a second-order belief predictor of the human player's action-policy.

Section II defines the search-and-rescue problem and sets up the human data collection pipeline and agent learning framework. Section III describes the necessary intervention scenarios observed in the human trials and the results of the agent training. Section IV concludes the main takeaways from the human trials, analyzes the challenges of agent learning, and discusses future work.

II. METHODOLOGY

In this section, we describe the search-and-rescue problem formalization, setup for the data collection from the human player trials in Minecraft and the curriculum-based reinforcement learning training setup for the artificial expert agent. The human trials are setup to easily extract first-order false beliefs, which can be used to further identify intervention scenarios. In parallel, we setup the expert agent learning to learn a value function, which acts as a second-order belief predictor. Online, the learned value function can be used to determine if an intervention is necessary by comparing the value of the player's action and the value of the optimal action. Both methods use theory of mind to predict player beliefs. Their difference is in the degree of separation from the player.

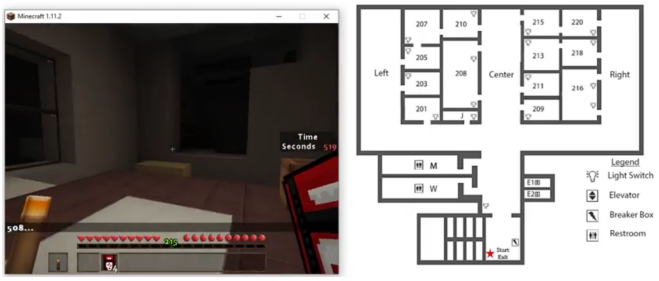


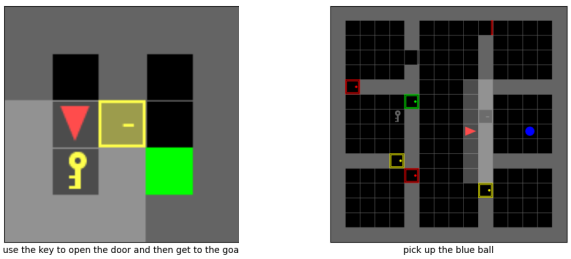
Fig. 2: The player has discovered a hole in the wall of an office that is not located of their blueprint map, influencing their trajectory planning. For instance, a player with a greedy triage strategy will look through the hole and immediately triage any and all victims seen.

A. Problem Definition

The goal of the agent, whether it be a software agent or human agent, is to maximize their score by rescuing victims. We frame the search-and-rescue challenge as a Partially Observable Markov Decision Process (POMDP), represented as a tuple (S, A, P, R, γ, O). S is our state space, where each state consists of the (x, y) position and θ rotation. The other sets are specific to either the human player trials or the expert agent training, so they are described in each section below.

B. Data Collection

For the human data collection, human players are tasked with searching through a simulated office building in Minecraft and rescuing the victims by triaging them. The human players are tasked with saving high-risk victims, which appear as yellow blocks, and low-risk victims, which appear as green blocks, with the goal of maximizing their score. Players have ten minutes to complete the challenge. The set of observations, O, consists of visual data from Minecraft in 1st person view and a static blueprint of the map of the office. Due to the disaster scenario that creates the search-and-rescue mission, there may be blockage where there is a connection on the blueprint or a hole in the wall that allows connection between rooms. Players complete three missions with different difficulty maps. Maps with more environmental disturbance, i.e. blockage or holes, are considered harder maps. High-risk victims die after five minutes, but are worth double the points of low-risk victims. In the human trials, the score feedback is on the screen. Thus, the rewards are sparse and only appear when the agent successfully triages victims. A is the set of four actions:



(a) Easy environment. (b) Hard difficulty environment.

(c) Human-difficulty environment.

Fig. 3: The above figure shows the easy environment used for training in figure 3a and the realistic environment used for training in figure 3c that is the same as the one used for human players in Minecraft. The hidden areas on the human environment have not been seen by the agent yet. The expert agent is trained on these environments as well as intermediate stages between difficulties.

rotate clockwise, rotate counterclockwise, move forward, and a toggle action. The toggle interacts with the environment, which includes opening or closing doors, turning on or off lights, and triaging victims. During gameplay, the player verbally describes their state of mind and plan of action in order to understand their internal states. A recording of gameplay and their commentary is used for analysis.

C. Artificial Agent Training Setup

The view seen by humans has been converted into a 2D grid environment in Minigrid. The two-dimensional grid simplifies the observation model for a learned agent. The set of observations, O, in Minigrid is an RGB image of the 2D grid environment. Note that in the minigrid environment, we add keys to open doors and limit the field of view to be similar to a human's field of view in Minecraft, creating partial observability. A is the set of four actions: rotate clockwise, rotate counterclockwise, move forward, and a toggle action. The rewards $R = \{0, 1\}$ are sparse, with reward earned when the toggle action is performed on a space with a victim or when locked doors are open. Reward is also given when the agent enters a new room, encouraging exploration. Note that exploration is implicit in human agents.

Our training uses proximal policy optimization (PPO), a form of a policy gradient method for reinforcement learning. PPO takes multiple episodes into account using stochastic gradient descent to perform policy updates. It has competitive performance on 2D platform games such as the ATARI games, making it ideal for use in a 2D Minigrid environment [10]. The algorithm has been modified from [11], [12]. We use our own convolutional neural network to develop the feature space and a fully connected neural network for the actor and critic, respectively, to do inference. Additionally, we use an LSTM's memory cell to represent internal states. The memory cell data is passed as a feature to the fully connected network.

The agent is trained in a series of environments with difficulty relative to the size of the map, that is, the larger maps are more difficult. The hard difficulty in figure 3c, shows the same layout as the environment in which the humans players were studied. The expert agent is trained starting with the least complex environment to the most complex environment in a curriculum learning fashion. Three sample environments are seen in figure 3.

III. RESULTS

In this section, we describe the results of the data collection from the human player trials in Minecraft and the curriculum-based reinforcement learning training for the expert agent. We identify first-order false beliefs found in the human trials and identify classes of intervention scenarios. In parallel, we learn a value function of the expert agent. The value function can act as a second-order belief predictor. Online, the learned value function can be used to determine if an intervention is necessary by comparing the value of the player's action and the value of the optimal action. Both methods use theory of mind to predict player beliefs. In the case of the expert agent, it is machine theory of mind due to the artificially-generated data.

A. Human Observational Data Results

From our observations, we can break down the interventions into the following categories: triaging strategy, memory, and terrain changes. Below we describe a series of observed scenarios that require intervention on behalf of an agent. These scenarios contain first-order false beliefs made by the human player. Note that we find that the interventions may be policy dependent, so noting the specific strategy of the human is necessary to make and discuss appropriate interventions.

1) Triaging Interventions: First, we consider the triaging strategy. Due to the complexity of the challenge, different triaging strategies may be optimal, but all strategies may yield suboptimal situations. From the observed human data, players either consider the high-risk (yellow) victims first or greedily triage all victims as they are found. Interventions regarding the victims are the simplest. For both strategies, if the yellow victims are in the player's field of view and the player does not immediately path to triage them then this indicates a false first-order belief so the agent should intervene. In a greedy strategy, a human will triage all victims

once seen. In this case, if the human does not triage a victim, yellow or green, which is in its field of view, the agent should intervene.

Note that the training strategy also affects the optimal trajectory. When following a greedy strategy and playing optimally, the player should not revisit any rooms. If the player must revisit a room, it is clear that they have missed a victim during their first pass. It could be the case that the player did not see the victim in one's field of view or observe the beeps, which creates a false first-order belief of the environment. In this case, there are two new types of interventions. The agent should intervene if the player does not view the entire room. The agent should also intervene if the agent receives observations such as beeps declaring victims nearby and does not react according to their triaging strategy. While the agent and player may not necessarily know what new information is provided by the beeps, the player should try to learn to use this information. This will result in a change in the anticipated trajectory of the player.

Not all of the yellow victims may be triaged in the greedy strategy. If the layout of the map is not favorable for the player to find the yellow victims, e.g. all the yellow victims are in the last room that the player looks in, the greedy strategy is suboptimal. The challenge is for the agent to determine that the greedy strategy is suboptimal. Suppose that a greedy player has 30 seconds left to triage the yellow victims, which take 15 seconds to triage, and there is one yellow victim remaining. If there is no observed data, field of view or otherwise, that determines the location of the victim, the agent should intervene so that the player switches strategies to ensure that no victims succumb to their injuries. In this case, the player had a first-order false belief about their environment with regard to the amount of time remaining. Note that it may be better to switch the strategies before 30 seconds remain. An example safe heuristic would tell the player to switch strategies if the amount of time remaining to run through each room, observe where the remaining high-risk victims are, and triage them is close to the time remaining. Of course, the previous heuristic does not take terrain changes into account.

2) On Searching Strategies: Optimal path planning strategies may not be reflected in the human data. However, we believe that the optimal path planning strategy is not a strategy in itself, but rather an accompaniment to the triaging strategy. That is, certain triaging strategies should be combined with certain searching strategies. These have already been described and the decisions made in searching are displayed in figure 4.

3) Memory Interventions: Memory interventions come from the basic assumption that redundant searching is suboptimal. When the player follows a memory-intensive searching strategy, such as Second Sweep, where the player first locates all the victims, triaging yellow victims greedily, and then going back during a second sweep and triaging green victims, there are more opportunities for the player to mess up, leading to a first-order false belief of the environment, and require intervention. Players may forget where the green

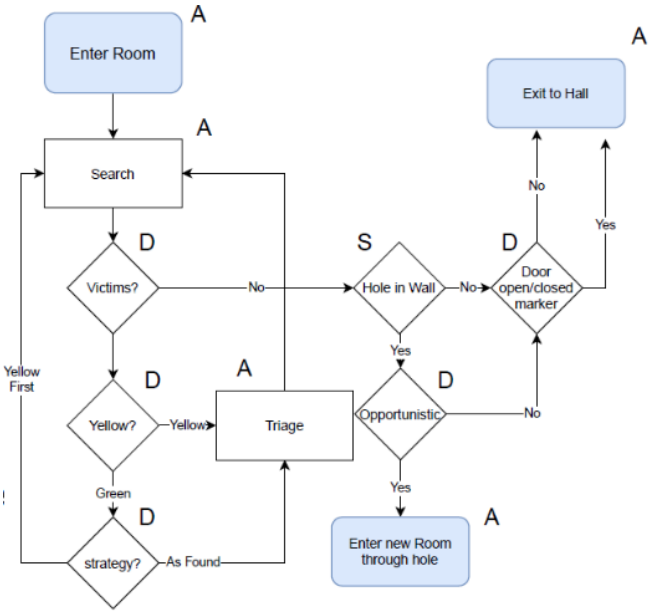
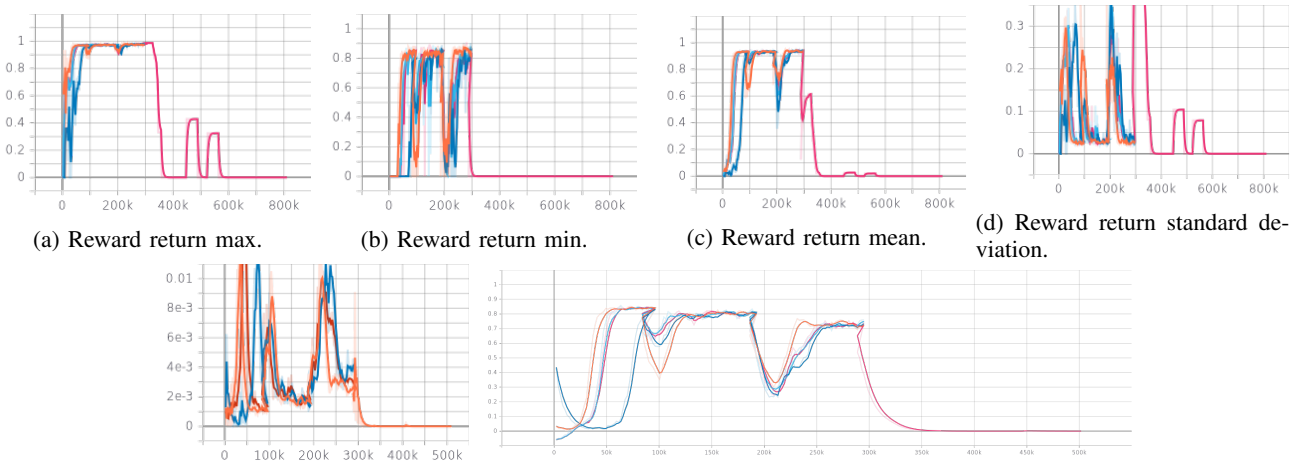


Fig. 4: Human observation data shows a series of fixed searching strategies based on the triaging strategy.

victims were and end up redundantly searching. Optimally, the agent can intervene and tell the player that they have already been in the room before or advise the player to go to an unvisited room. Another strategy for the agent is to intervene earlier and develop a strategy in the player that prevents forgetting and redundant searching. The agent can intervene in rooms and tell the player to interact with blocks to mark the area. This may be as simple as turning a light on when entering the room or turning off the light and leaving the door open when the room has been properly triaged. The interactions with the environment acts as a signal which creates an additional feature which helps foster correct first-order beliefs regarding the environment.

4) Terrain and Interventions: Define terrain changes as blockage or holes that appear in the environment. Terrain changes require deviation from intended trajectories. Terrain changes may cause the player to backtrack in order to reach an unseen area of the environment or require more exploration in order to reach unseen areas. This can again be seen as a memory problem. The agent needs to be able to observe blockages or holes and use this information to update the trajectory planning strategy for the player. For example, the agent should remember another way to reach a room or suggest a way to explore the environment to find such a path when blockages are seen. Terrain changes can also affect the player's first-order belief in one's location. When there is a hole, the player may choose to explore this and take advantage of known terrain changes. This exploration may end up being suboptimal if it slows down search overall or necessary if that is the only way to reach a room. All these factors depend on the triaging strategy. The important part is for the agent to be able to take the environment changes and triaging strategy into consideration and use them to help



(a) Reward return max. (b) Reward return min. (c) Reward return mean. (d) Reward return standard deviation.

(e) Value function loss. (f) Value function.

Fig. 5: Reinforcement training graphs over 3 seeds. The X-axis shows the frame number. The graphs show that as environment gets more complex, the expert agent is able to still effectively complete the task, but performance decreases slightly. Once the environment's complexity is difficult for humans, the agent is no longer able to complete the task.

suggest changes to the triaging strategy.

B. *Artificial Agent Experiments*

We train an artificial expert agent in order to learn a value function which can be used to determine when interventions are required. Training in the easy to human-level difficulty environments, such as the environments seen in 3, can be seen in figure 5 with three seeds for comparison. Figure 5f shows that the value function diminishes as the agent is trained in more challenging environments until ultimately not being able to complete the task in the human-difficulty environment. The value loss output in figure 5e concludes the same result: training is successful until the environment complexity is too great. The maximum reward return in figure 5a shows that optimal performance can occur in most environments. The reward minimum return in figure 5b yields a lower bound on good performance in the first three environments. The reward standard deviation in figure 5e shows that beginning training in new environments is more difficult as the complexity increases. We found that despite curriculum-based learning, training in the human-difficulty environment (figure 3c) yielded random performance even with training for up to 10 million game frames.

IV. CONCLUSIONS

With respect to the human data study, the main intervention strategies can be broken down into two separate problems. The first is triaged-based: how can the agent determine when the player should change one's triaging strategy. This recognizes a player's first-order false beliefs. The second is how can the agent create and make interventions to the player's memory. This encourages fostering first-order correct beliefs in the player. From here, the triage-based strategy can be recognized based on the trajectory. This information should be passed as a feature to the intervention agent in order to make the binary decision whether an intervention is required. The memory intervention requires introspection in order to correct the current saved internal state. For the learned agent, the smaller environments were able to be solved, but performance falls off as the grid increases in size.

With respect for the second task of this paper, the learned expert agent's value function output analyzed in larger environments shows that the agent is not able to distinguish short term goals. Even with an LSTM, the memory of the agent appears to be too small. The issue with training in larger environments is that the reward signal is sparse. In future work, this will be addressed by adding short-term rewards for intermediate goals. Training with larger memory requires much more RAM than was available for the experiments. Future work will also include an analysis on human data to see if the learned value function of the expert intervention agent is able to correctly identify second-order beliefs of the player; that is, the analysis will yield if the interventions identified online are valid.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. The authors thanks Rachel Burcin, John Dolan, and all of Carnegie Mellon University's Robotics Institute Summer Scholars (RISS) program. Special thanks to Huao Li, Vidhi Jain, and Tejus Gupta for their guidance and direction.

REFERENCES

[1] D. Premack and G. Woodruff, "Does the chimpanzee have a theory of mind?" *Behavioral and Brain Sciences*, vol. 1, no. 4, p. 515–526, 1978.

[2] M. Shum, M. Kleiman-Weiner, M. L. Littman, and J. B. Tenenbaum, "Theory of minds: Understanding behavior in groups through inverse planning," *CoRR*, vol. abs/1901.06085, 2019. [Online]. Available: http://arxiv.org/abs/1901.06085

[3] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," *CoRR*, vol. abs/1709.04326, 2017. [Online]. Available: http://arxiv.org/abs/1709.04326

[4] J. Oh, F. Meneguzzi, and K. Sycara, "Probabilistic plan recognition for proactive assistant agents," *Plan, Activity, and Intent Recognition: Theory and Practice*, pp. 275–288, 03 2014.

[5] J. Oh, F. Meneguzzi, K. Sycara, and T. Norman, "Prognostic normative reasoning in coalition planning," vol. 2, 01 2011, pp. 1233–1234.

[6] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine theory of mind," *CoRR*, vol. abs/1802.07740, 2018. [Online]. Available: http://arxiv.org/abs/1802.07740

[7] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *CoRR*, vol. abs/1806.06877, 2018. [Online]. Available: http://arxiv.org/abs/1806.06877

[8] A. Nematzadeh, K. Burns, E. Grant, A. Gopnik, and T. L. Griffiths, "Evaluating theory of mind in question answering," *CoRR*, vol. abs/1808.09352, 2018. [Online]. Available: http://arxiv.org/abs/1808.09352

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," vol. 60, 01 2009, p. 6.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[11] L. Willems, "rl-starter-files," https://github.com/lcswillems/rl-starter-files, 2020.

[12] ——, "torch-ac," https://github.com/lcswillems/torch-ac, 2020.

Robotic Herding by Collision Avoidance for Robotic Swarm

Benjamin Kazules[1], Dr. Katia Sycara[2], and Wenhao Luo[3]

***Abstract*— The purpose of this paper is to develop a simple algorithm to allow a number of dog robots to defend a protected zone from adversarial sheep robots, and to provide experimental validation for the algorithm in various configurations. The algorithm prioritizes the sheep robots based on distance from the center of the protected zone and assigns dog robots to intercept them. We allow a number of parameters to vary, including numbers of sheep and dog robots as well as relative speeds. The experiments demonstrate that any number of dog robots greater than or equal the number of sheep robots is sufficient to guard the protected zone. In addition, if certain assumptions can be made about the organization of the sheep robots, into a formation for example, fewer dog robots are needed to fully guard the protected zone.**

***Index Terms*— Agent-based systems, autonomous agents, multi-robot systems, swarms.**

I. INTRODUCTION

Robotic herding is very important to the military as robotic swarms are becoming more of a reality. The United States military maintains protected airspace around every installation and in many operational environments. In order to effectively defend these areas from potentially adversarial robotic swarms, algorithms must be developed to guarantee successful herding and defense. In this paper we develop one such algorithm and provide examples of different scenarios where it can be expected to succeed or fail.

II. PROBLEM STATEMENT

The problem is as follows: given N sheep robots and M dog robots, control the dog robots to herd the sheep robots away from a certain region, called the protected zone. For the time being, no assumptions are made about the sheep robot behavior or intentions. The problem is straightforward for the case of one dog and one sheep robot, which naturally extends to N dog and N sheep robots. We begin to examine further scenarios, such as different numbers of sheep and dog robots.

III. SIMULATION AND ALGORITHM DESIGN

A. Algorithm

The algorithm is made up of three main parts that are iterated as the simulation runs. It first prioritizes the sheep robots based on their distance from the protected zone, with the closest sheep robot being the highest priority. Next, the dog robots are assigned to defend against individual sheep robots in priority order. For each sheep robot, the closest unassigned dog robot is assigned. The final step controls the dog robots to intercept their assigned sheep robots by moving toward the midpoint between the sheep robot and the protected zone. The assignments are only updated every twenty time steps in the simulations, and any extra dog robots are sent to a staging area around the protected zone.

B. Simulation

The simulations were run in MATLAB using the robot dynamic model as in [1] and barrier certificates to avoid collisions [2]. The sheep robots were controlled simply by individually moving toward a common goal. In the formation simulations, the sheep robots used a swarm formation controller from [1] with the leader of the formation moving toward a goal.

IV. RESULTS

The simulations overall showed that any number of dog robots greater than or equal to the number of sheep robots will successfully defend the protected zone given that the sheep robots start far enough away. In real world scenarios this will virtually always be the case, as the dog robots would already be at the protected zone before the sheep robots approach. However, if the dog robots have a high enough speed, they may momentarily ward off the sheep robots with some success. This seems to always end in failure if allowed to continue to its conclusion, although if the protected zone is merely in the flight path of the sheep robots then fewer dog robots can likely be successful. In addition, if certain assumptions are made about the sheep robots such as a rigid formation with a single leader, then success is also possible with fewer dog robots, even as little as one. This is particularly useful for when the sheep robots are not strictly adversarial and the protected zone is merely in their original flight path.

A. One Dog, One Sheep

The case with one dog robot and one sheep robot is simple, with the dog robot moving to intercept the sheep robot. Once the dog robot has done so, we reach a stable state where the dog robot is directly in between the sheep robot and the protected zone (Fig. 1). In this state, the dog robot is able to immediately block any attempt of the sheep robot to get around it unless the sheep robot is significantly faster then the dog robot. This simulation does not directly model such a ”smart” sheep robot, but it is clear that if the dog robots

[1]Benjamin Kazules is a cadet at the United States Air Force Academy, CO 80840, USA C21Benjamin.Kazules@afacademy.af.edu

[2]Dr. Katia Sycara is director of the Advanced Agent - Robotics Technology Lab in the Carnegie Mellon University Robotics Institute 5000 Forbes Avenue, Pittsburgh, PA 15213, USA sycara@andrew.cmu.edu

[3]Wenhao Luo is a PhD student at the Carnegie Mellon University Robotics Institute 5000 Forbes Avenue, Pittsburgh, PA 15213, USA luo@cmu.edu

are at least as fast as the sheep robots, then this case leads to success.

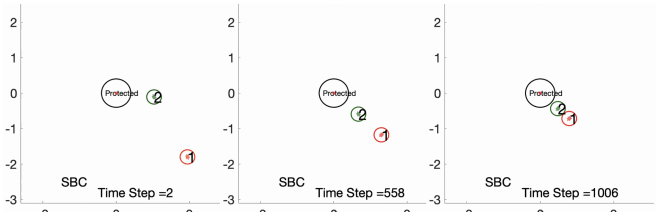


Fig. 1. One dog, one sheep simulation

B. N Dogs, N Sheep

The case with equal numbers of dog and sheep robots is a simple extension of the one on one case as long as there is no significant crowding around the protected zone (Fig. 2). This crowding is not well handled by the simple algorithm used in this paper, but potentially could be in the future. In practical scenarios the protected zone would be large enough that this crowding would not be a problem however, and if not then the dog robots could instead be made to form a tight ring around the small protected zone, avoiding the problems that arise from cluttering.

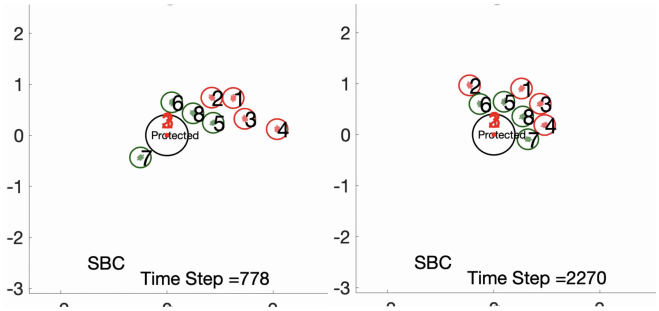


Fig. 2. 4 dogs, 4 sheep simulation

C. One Dog, Multi-Sheep Formation

If we assume a rigid formation or similarly any strict leader-follower swarm behavior on the sheep robots, then the dog robot controller may be simply adjusted to target the leader and push it away from the protected zone to a distance far enough that no robot in the swarm enters the zone. The examples we have assume the formation does not break (Fig. 3). While this is not realistic, once the formation breaks the situation becomes identical to the N dogs, M sheep case, which is dealt with in the next section. If the sheep robots are passing through to some other location like a flock of birds might on a migratory path, it may be possible to use no more than a single dog robot to herd them around the protected zone (Fig. 4). In general, the more assumptions are made of the sheep robots and the more specific they are, the higher the chance a smaller number of dog robots will be required.

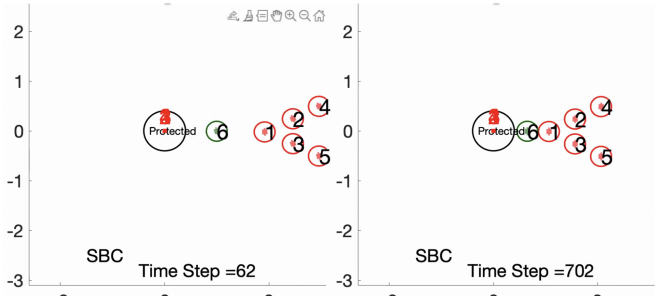


Fig. 3. One dog, multi-sheep formation attacking

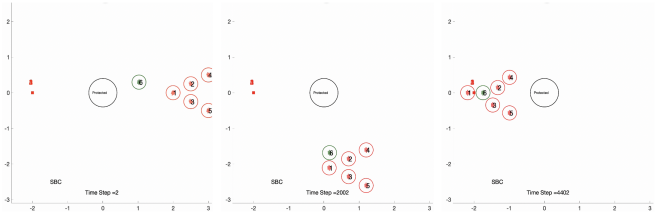


Fig. 4. One dog, multi-sheep formation passing through

D. N Dogs, M Sheep

This case has two distinct sub-cases, where the number of dog robots N is either greater than or less than the number of sheep robots M. In the case with $N > M$, the M dog robots that are closest to the sheep robots act in the same way as the N dog, N sheep case. This means that the same guarantees will be in place. The extra dog robots are sent to guard the protected zone where it is left open, but can be made to do anything else that may be useful (Fig. 5). In the case with $N < M$, the dog robots can be successful for a period of time in a direct assault, but if the simulation is allowed to run long enough, it seems to always end in failure (Fig. 6). More dog robots will be more effective even if they are less than the sheep robots, but having at least half the number seems most important. In addition, increased speed of the dog robots also makes them more effective in defense, as they are better able to split their effort between different sheep robots. There may be a point at which increased speed can deliver a theoretical guarantee of success in the long run, but the simulations do not support this as of yet. While fewer dog robots seem to be entirely unable to defend the protected zone from directly adversarial sheep robots, it is often possible for them to defend against larger swarms of sheep robots passing through the protected zone (Fig. 7).

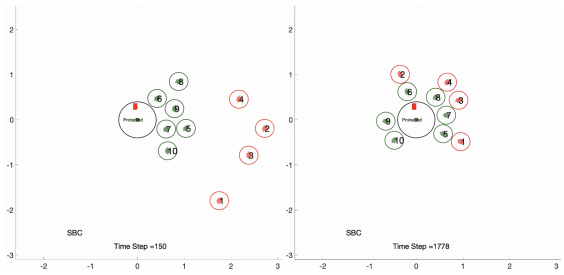


Fig. 5. 6 dogs, 4 sheep simulation

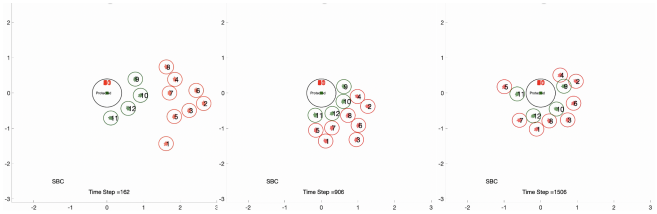


Fig. 6. 4 dogs, 8 sheep attacking simulation

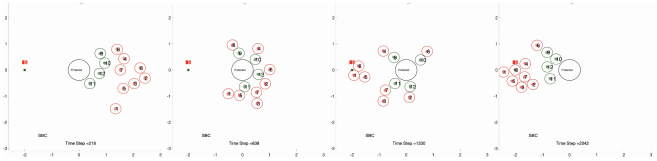


Fig. 7. 4 dogs, 8 sheep passing through simulation

V. CONCLUSIONS

The overall results show that as long as the number of dog robots is equal to or greater than the number of sheep robots and there are no extreme circumstances such as crowding or the sheep robots suddenly appearing up close, the algorithm will be successful in defending the protected zone. If assumptions are made about the sheep robots movement, then the number of dog robots may often be decreased while still effectively herding the sheep robots away. In addition, if the sheep robots are only attempting to pass through the protected zone to get to an outside goal, the number of dog robots may similarly be decreased. These results require proofs to be taken as formal guarantees, but the simulations in this paper support them experimentally.

ACKNOWLEDGMENT

The authors would like to thank Carnegie Mellon University, the Robotics Institute, and the United States Air Force Academy for the opportunity to conduct this research.

REFERENCES

[1] K. S. Sasanka Nagavalli, Nilanjan Chakraborty, "Automated sequencing of swarm behaviors for supervisory control of robotic swarms," *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[2] A. D. A. Li Wang and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, June.

[3] K. K. Aditya A. Paranjape, Soon-Jo Chung and D. H. Shim, "Robotic herding of a flock of birds using an unmanned aerial vehicle," *IEEE Transactions on Robotics*, vol. 34, no. 4, 2018.

Safe Planning and Control via Constrained ILQR and Robust Tube MPC

Shivesh Khaitan[1], Qin Lin[2] and John M. Dolan[2]

***Abstract*— Safe planning is critical for self-driving to generate collision-free trajectories. Recently, a safe planning framework with safety guarantee using CILQR and reachability has been proposed. The significance of this approach is addressing the prediction uncertainty of surrounding vehicles while trajectory planning. However, the uncertainty from the control level still poses challenges to safe execution of the trajectory. For example, the uncertainty of the dynamic model of the ego-vehicle and disturbances in actuators, which are common, may jeopardize the safety of the ego-vehicle. Such uncertainty can be dealt with using a robust controller which can generate control commands taking the uncertainty into consideration. Several variants of Robust MPC have been extensively studied for trajectory planning under uncertainty. However, the available literature does not consider Robust MPC for a low-level controller in non-linear systems capable of obstacle avoidance. In this paper, the existing safe planning framework is extended with a low-level robust tube MPC, which can avoid dynamic obstacles while closely tracking the path planned by CILQR and ensuring safety guarantees in the presence of ego-vehicle model and control uncertainty. We also demonstrate safety, effectiveness and real-time performance of our framework in the CARLA simulator.**

***Index Terms*— Autonomous Vehicle Navigation, Collision Avoidance, Motion and Path Planning, Optimization and Optimal Control**

I. INTRODUCTION

Self driving cars are expected to eliminate the possibility of accidents. Thus, the algorithms used in self-driving should guarantee safety. A huge section of motion planning autonomous driving research only deals with planning and controls in a deterministic environment. But safety, which is the key driving factor for introducing autonomous vehicles, is not guaranteed in such cases. In recent decades, motion planning research has been extended to consider uncertainty and safety is able to be guaranteed for bounded disturbances. Still, there does not exist a framework that comprehensively assures safety across planning, and control layers in uncertain environments.

One problem arises from the highly uncertain trajectories of surrounding vehicles due to sensing, localization, maneuver or intention uncertainties, etc. This has been studied and successfully dealt with in [1]. However, similar to the uncertainty of the environment, the uncertainty arising due to the error in the assumed ego-vehicle's model and control cannot be ignored. This uncertainty can arise due to:

1) Error in the assumed dynamic ego-vehicle model.
2) Disturbance in the actuators while trying to achieve the control commands.
3) Measurement noise leading to errors in state estimation.

Safety of the autonomous system cannot be guaranteed unless such noise is taken into consideration. [2] addresses a similar problem but only has probabilistic guarantee. Model Predictive Control (MPC) [3]–[6] is one of the most successful approaches for system controls. The basic idea of MPC is to repeatedly solve optimization problems on-line to find an optimal input to the controlled system. Several MPC variants have been studied in recent decades which are robust to uncertainty. Majorly, the robust algorithms can be divided into two classes of approaches: min-max techniques and robust tube MPC approaches.

Min-max techniques optimize the worst-case performance of the controller with respect to bounded uncertainties. For a comprehensive literature review on min-max MPC techniques, readers are referred to [7]. A general drawback of this technique is its worst case consideration. This exponentially increases the computational requirements, which is not suitable for real time implementation.

The robust tube-based MPC techniques guarantee to keep the actual state within an invariant tube around the nominal MPC trajectory. They work by complementing the nominal MPC with a feedback controller which tries to keep the actual state within the tube by tracking the nominal trajectory. Tube-based approaches have been extensively studied in the literature [8]–[12]. However, these do not consider the presence of obstacles, in which non-convex state constraints need to be satisfied. [13] and [14] have proposed techniques for collision avoidance, but [13] deals with static obstacles only, and [14] uses invariant set computation for obstacles, which incurs high computation costs.

In this paper, we propose a computationally efficient unified safe planning and control framework for a non-linear system with the ability to avoid moving obstacles. The aspect that sets our method especially apart from existing methods is that they mostly use robust control for high-level trajectory planning, while we exploit the existing tube MPC as a low-level control for a high-level trajectory planner. The main motivations for this are:

1) Non-linearity in dynamics can be linearised in a low-level controller using approximation since the reference point would be very close to the actual state. This is not the case in high-level robust trajectory planners.

[1]Shivesh Khaitan is with the Department of Computer Science & Engineering, Manipal Institute of Technology, Manipal, Karnataka, India shivesh.khaitan@learner.manipal.edu

[2]Qin Lin and John M. Dolan are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, the USA {qinlin,jdolan}@andrew.cmu.edu

2) The moving obstacles can be considered static across the whole planning horizon in a control cycle since the total simulation time is minimal.

We showcase the framework by adopting the robust tube controller proposed in [8] to guarantee control-level safety for Safe Planning [1]. Robust tube control, however, cannot directly deal with moving obstacles which result in a non-convex free space. For this, we propose convexification of the space around the ego-vehicle, which can provide the controller with an obstacle-free convex region, with the actual state of the ego-vehicle in its interior. This convex constraint can then be tightened for robust control, as we will discuss in section III. We make use of IRIS (Iterative Regional Inflation by Semidefinite programming) [15] for convexification of the space. Figure 1 shows a high-level overview of the proposed framework.

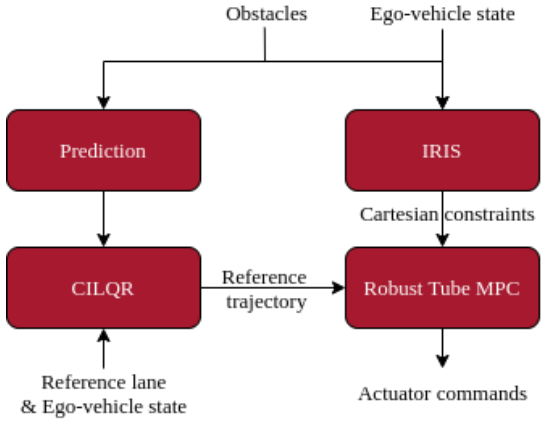


Fig. 1: Overview of the framework.

The rest of this paper is organized as follows. Section II contains necessary background of key methodologies. Section III is about the formulation of the vehicle dynamic model, cost function and constraints design. Section IV presents the experimental results. The conclusion is in section V.

II. METHODOLOGY

In this section, we will go through key techniques used including short and long-term prediction using reachability analysis and an adaptive filter, constrained ILQR optimization, convexification using IRIS and constrained linear robust tube MPC.

A. Prediction of dynamic obstacles

The state of the moving obstacles in the environment needs to be predicted in each planning loop to generate obstacle-free trajectories. This is achieved using the short-term and long-term prediction model from [1]. It proposes a combination of a safety-oriented short-term planner and an efficiency-oriented long-term planner.

The short-term prediction considers the uncertainty of a target vehicle's state (e.g., sensor disturbance or localization error) and the uncertainty of control actions over the short-term prediction horizon under a kinematically feasible but possibly non-deterministic assumption. The reachable state of the target vehicle is projected to the sub-space in the inertial frame for the min/max longitudinal and lateral positions. The long-term predictor only predicts the target vehicle's single position (i.e., particle) without considering uncertainty. For a detailed explanation of the short and long term predictions, readers are referred to [1].

B. Constrained Iterative Linear Quadratic Regulator (CILQR)

The robust tube low-level controller requires a reference trajectory which it should track. To compute the reference trajectory, we make use of CILQR [16], [17]. An obstacle-free motion planning problem can be formulated as a standard ILQR problem with nonlinear system dynamics:

$$\min_{\boldsymbol{U}} \quad J = \sum_{k=0}^{N-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k) + l_f(\boldsymbol{x}_N) \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{2}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are the state and the control input at time step k and $\boldsymbol{x}_N$ is the final state. Eq. 2 is the system dynamics constraint, which is a transition function mapping state and control at step k to state at step $k+1$. $\boldsymbol{U} := \{\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_{N-1}\}$ is the control sequence, l and l_f are the cost functions.

Since the standard LQR only solves optimization problems with quadratic cost and linear systematic constraints, this problem can be reformulated. By linearizing the systematic constraint at multiple points, we can relax the nonlinearity of the ILQR problem into the linear problem required by LQR. The steps of ILQR are listed below.

1) Start with a feasible initial guess $\hat{\boldsymbol{u}}$ and obtain $\hat{\boldsymbol{x}}$ using the dynamic model. A common way is using a zero initialization. Note that feasibility of the initial guess is important in practice. The users can either do a sampling in the beginning or start the planning only when the zero initialization is feasible.
2) Calculate the derivatives of the dynamics and the cost function about the trajectory.
3) Run an LQR backward pass to obtain $\delta\boldsymbol{u}^*$. For an ill-conditioned matrix, we increase λ and restart the backward pass, otherwise we reduce λ. The details of designing appropriate factors of increasing and decreasing can be found in [18].
4) Run forward pass and initially set $\alpha = 1$ in $\delta\boldsymbol{u} = \alpha\boldsymbol{k} + K\delta\boldsymbol{x}$ to compute a new nominal sequence. If the cost does not converge, decrease α and restart the forward pass.

ILQR has the drawback of its constraint-free nature, which makes it unsuitable for collision avoidance problems. The CILQR algorithm offers the inclusion of different constraints into the objective function through barrier functions. Ideally, a barrier function serves as an indicator giving a huge penalty to constraint violation and low cost to satisfied constraints.

Constraints can be generalized into two categories by linearity. First, any nonlinear constraints can be converted to linear constraints via a second-order Taylor Expansion. Then, a barrier function is applied and quadratized. Eq. 3 and Eq. 4 demonstrate this process. The quadratized linear barrier function can now be incorporated into the ILQR algorithm.

An exponential barrier function is defined as

$$b_k(g(\boldsymbol{x}_k)) = q_1 \exp(q_2 g(\boldsymbol{x}_k)) \tag{3}$$

Its Jacobian and Hessian are derived as

$$\begin{aligned}\nabla b &= q_1 q_2 \exp\left(q_2 g(\boldsymbol{x}_k)\right) \nabla g(\boldsymbol{x}_k) \\ \nabla^2 b &= q_1 q_2 \exp\left(q_2 g(\boldsymbol{x}_k)\right)(q_2 \nabla g(\boldsymbol{x}_k) \nabla g(\boldsymbol{x}_k)^T + \nabla^2 g(\boldsymbol{x}_k))\end{aligned} \tag{4}$$

where $g(\boldsymbol{x})$ is the constraint function at time step k.

C. Obstacle-free convex state space constraint

As the robust tube MPC requires convex constraints, we use IRIS [15] to get a convex obstacle-free region containing the ego-vehicle. IRIS alternates between two convex optimizations (A) a quadratic program that generates a set of hyperplanes to separate a convex region of space from the set of obstacles and (B) a semidefinite program that finds a maximum-volume ellipsoid inside the polytope intersection of the obstacle-free half-spaces defined by those hyperplanes. Given the position of the ego-vehicle as an initial seed point in space, around which the first ellipsoid is constructed, IRIS grows the ellipsoid greedily at every iteration until it reaches a local fixed point. The final set of separating hyperplanes forms a convex polytope, which we use as our convex region of obstacle-free space.

D. Constrained Linear Robust Tube MPC

A constrained linear robust tube MPC solves the optimization problem defined as:

$$\min_{\boldsymbol{U}} \quad J = \sum_{k=0}^{N-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k) + l_f(\boldsymbol{x}_N) \tag{5}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{w_k} \tag{6}$$

$$\boldsymbol{x} \in \mathbb{X} \tag{7}$$

$$\boldsymbol{u} \in \mathbb{U} \tag{8}$$

where $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times n}$ are dynamics and input matrices respectively. $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ and $\boldsymbol{w}_k$ are the state, control input and disturbance respectively at time step k and $\boldsymbol{x}_N$ is the final state. $\mathbb{U} \subset \mathbb{R}^m$ is compact, $\mathbb{X} \subset \mathbb{R}^n$ is closed, and each set contains the origin in its interior. The disturbance $\boldsymbol{w}$ is assumed to be bounded as $\boldsymbol{w} \in W$ where W is compact and contains the origin (but may not have an interior).

Eq. 6 is the system dynamics constraint, which is a transition function mapping state, control and disturbance at step k to state at step $k+1$. $\boldsymbol{X} \coloneqq \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ is the optimal state sequence. l and l_f are the cost functions.

Let $\bar{\boldsymbol{X}} \coloneqq \{\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{x}}_1, \cdots, \bar{\boldsymbol{x}}_N\}$ be the optimal states sequence and $\bar{\boldsymbol{U}} \coloneqq \{\bar{\boldsymbol{u}}_0, \bar{\boldsymbol{u}}_1, \cdots, \bar{\boldsymbol{u}}_{N-1}\}$ be the optimal control sequence for the above problem when disturbance $\boldsymbol{w}$ is ignored. Let $\boldsymbol{K} \in \mathbb{R}^{m\times n}$ be such that $A_K = A + BK$ is stable. Let $\boldsymbol{Z}$ be a disturbance invariant set for the controlled uncertain system $x^+ = A_K\boldsymbol{x} + \boldsymbol{w}$, therefore satisfying $A_K Z \oplus W \subseteq Z$, where $\oplus$ denotes Minkowski set addition. Then, if $\boldsymbol{x}_0 \in \bar{\boldsymbol{x}}_0 \oplus \boldsymbol{Z}$, a sub-optimal control is given by $\boldsymbol{u} = \bar{\boldsymbol{u}} + K(\boldsymbol{x} - \bar{\boldsymbol{x}})$ and $\boldsymbol{x}^+ \in \bar{\boldsymbol{x}}^+ \oplus Z$ for all $\boldsymbol{w} \in W$ where $\boldsymbol{x}^+ = A\boldsymbol{x} + B\boldsymbol{u} + \boldsymbol{w}$ and $\bar{\boldsymbol{x}}^+ = A\bar{\boldsymbol{x}} + B\bar{\boldsymbol{u}}$. This sub-optimal control $\boldsymbol{u}$ keeps the states $\boldsymbol{x_k}$ of the uncertain system close to the states $\bar{\boldsymbol{x}}_{\boldsymbol{k}}$ of the nominal system with no disturbance when Eq. 11 and Eq. 12 are satisfied. The sub-optimal control guarantees that $\boldsymbol{x_k}$ will always be inside the convex constraint set $\mathbb{X}$. Thus instead of optimizing for Eq. 5, we solve for the nominal problem without disturbance as:

$$\min_{\bar{\boldsymbol{U}}} \quad \bar{J} = \sum_{k=0}^{N-1} l(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k) + l_f(\bar{\boldsymbol{x}}_N) \tag{9}$$

$$\text{s.t.} \quad \bar{\boldsymbol{x}}_{k+1} = A\bar{\boldsymbol{x}}_k + B\bar{\boldsymbol{u}}_k \tag{10}$$

$$\bar{\boldsymbol{x}} \in \mathbb{X} \ominus Z \tag{11}$$

$$\bar{\boldsymbol{u}} \in \mathbb{U} \ominus KZ \tag{12}$$

$$\boldsymbol{x_0} \in \bar{\boldsymbol{x}}_{\boldsymbol{0}} \oplus Z \tag{13}$$

The applied robust control is then $\boldsymbol{u} = \bar{\boldsymbol{u}} + K(\boldsymbol{x} - \bar{\boldsymbol{x}})$. For a detailed proof and study of feasibility and stability of the above controller, readers are referred to [8].

III. PROBLEM FORMULATION

In this section, we will go through the vehicle dynamic model, the cost function and constraints design.

A. System Dynamics

The model used is a kinematic bicycle model as shown in Figure 2; however, it can be replaced by a more complex dynamic model. Finite difference techniques can be used for the Jacobian and Hessian calculations. The control input is $\boldsymbol{u}_k = [a_k, \delta_k]^T$, where a_k and δ_k are the acceleration and steering angle, respectively. Eq. 14 represents the transition function with non-zero steering angle. The horizontal increment is calculated by $\int_0^l \cos(\theta_k + \kappa s) ds$ and the vertical increment is calculated by $\int_0^l \sin(\theta_k + \kappa s) ds$. $\kappa = \frac{\tan(\delta)}{L}$ is the curvature, L is the vehicle length, and $l_k = v_k \Delta t + \frac{1}{2} a \Delta t^2$ is the distance travelled at time k with discretization step Δt.

$$\begin{aligned} p_{x,k+1} &= p_{x,k} + \frac{\sin(\theta_k + \kappa l_k) - \sin\theta_k}{\kappa} \\ p_{y,k+1} &= p_{y,k} + \frac{\cos\theta_k - \cos(\theta_k + \kappa l_k)}{\kappa} \\ v_{k+1} &= v_k + a\Delta t \\ \theta_{k+1} &= \theta_k + \kappa l_k \end{aligned} \tag{14}$$

However, when δ is zero, κ will be zero, which will cause a numerical issue in Eq. 14. Hence, a different update shown in Eq. 15 is used.

$$\begin{aligned} p_{x,k+1} &= p_{x,k} + (v_t dt + \frac{1}{2} a \Delta t^2)\cos(\theta_k) \\ p_{y,k+1} &= p_{y,k} + (v_t dt + \frac{1}{2} a \Delta t^2)\sin(\theta_k) \\ v_{k+1} &= v_k + a\Delta t \\ \theta_{k+1} &= \theta_t \end{aligned} \tag{15}$$

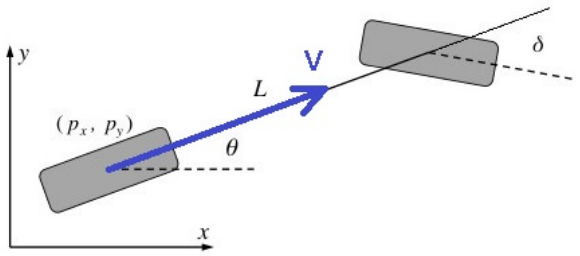


Fig. 2: Vehicle Kinematic Bicycle Model

B. Objective Function for CILQR

The general definition of the objective function in Eq. 1 can be made specific in Eq. 16. The terms in the summation represent the control effort cost, the reference position tracking, the reference velocity tracking, and the constraints cost (i.e., obstacle avoidance). c_N is the end state cost.

$$J = \sum_{k=0}^{N-1} \left(c_k^u + c_k^{ref} + c_k^{vel} + c_k^{con} \right) + c_N \tag{16}$$

1) control effort cost: The penalty for large acceleration and the steering angle with corresponding weights are shown in Eq. 17.

$$c_k^u = \boldsymbol{u}_k^T \begin{bmatrix} w_a & \\ & w_\delta \end{bmatrix} \boldsymbol{u}_k \tag{17}$$

2) reference tracking and velocity cost: The reference tracking term assigns a cost based on the distance to the closest point of the reference trajectory. The velocity cost penalizes the ego-vehicle for the difference between its velocity and the reference velocity. The combined cost is written in a matrix form in Eq. 18, where $\Delta \boldsymbol{x}_k$ is the difference between the ego-vehicle state and the reference state.

$$\begin{aligned} \Delta \boldsymbol{x}_k &= \boldsymbol{x}_k - \begin{bmatrix} p_{x,k}^{ref} & p_{y,k}^{ref} & v_k^{ref} & 0 \end{bmatrix}^T \\ c_t^{ref} + c_t^{vel} &= \Delta \boldsymbol{x}^T \begin{bmatrix} w_{ref} & & & \\ & w_{ref} & & \\ & & w_{vel} & \\ & & & 0 \end{bmatrix} \Delta \boldsymbol{x} \end{aligned} \tag{18}$$

3) constraint cost: All inequality constraints can be expressed in a negative null form shown in Eq. 19 in which x_{lim} is the maximum or minimum boundary value and f(x) is some sort of function on the decision variable.

$$g(x) = x_{lim} - f(x) \leq 0 \tag{19}$$

For linear constraints like acceleration and steering limits, we can write them as for instance:

$$g(u) = u - u_{max} \leq 0 \tag{20}$$

Then, a barrier function can be used as stated in Eq. 3.

For the obstacle avoidance term, we use a geometric collision check as the inequality constraint and the problem is formulated as a geometry-based cost function. Obstacles are formulated as ellipses with major and minor axes adjusted for the ego-vehicle's shape. The inequality constraint is shown in Eq. 21. θ_k^{obs} is the predicted heading angle of the obstacle at time k, a and b are semi major and minor axes' lengths. Δx_k and Δy_k are the relative longitudinal and lateral distance between the ego-vehicle and the predicted obstacle, respectively. The corresponding Jacobian and the Hessian of this term for the LQR backward pass can be found in Eq. 4.

$$\begin{aligned} R_k &= \begin{bmatrix} \cos(\theta_k^{obs}) & -\sin(\theta_k^{obs}) \\ \sin(\theta_k^{obs}) & \cos(\theta_k^{obs}) \end{bmatrix} \\ T_k &= R_k \begin{bmatrix} \frac{1}{a^2} & \\ & \frac{1}{b^2} \end{bmatrix} R_k^T \\ g([\Delta x_k, \Delta y_k]) &= 1 - [\Delta x_k, \Delta y_k] T_k [\Delta x_k, \Delta y_k]^T \leq 0 \end{aligned} \tag{21}$$

C. Objective Function and Constraints for Robust Tube MPC

The general definition of the objective function in Eq. 5 can be made specific in Eq. 22. Each term in the summation represents the control effort cost, the reference position tracking and the reference velocity tracking. c_N is the end state cost.

$$J = \sum_{k=0}^{N-1} \left(c_k^u + c_k^{ref} + c_k^{vel} \right) + c_N \tag{22}$$

A detailed description of the individual cost terms can be found in the previous section.

The constraints for MPC have to be updated in each control loop. The steps for computing the constraints are listed below.

1) Compute the disturbance invariant set $\boldsymbol{Z}$.
2) Using the instantaneous position of the ego-vehicle as the seed-point for IRIS, obtain an obstacle-free polytopic region P as:

$$A\boldsymbol{p}_x \leq b \tag{23}$$

$$A\boldsymbol{p}_y \leq b \tag{24}$$

3) Obtain the nominal state-space constraints, $\mathbb{X}$, by appending to P the polytope of velocity and orientation constraints:

$$-v_{max} < \boldsymbol{v} < v_{max} \tag{25}$$

$$-\pi < \boldsymbol{\theta} < \pi \tag{26}$$

4) Obtain the nominal control constraints, $\mathbb{U}$ by computing the polytope of acceleration and steering constraints:

$$-a_{max} < \boldsymbol{a} < a_{max} \tag{27}$$

$$-\delta_{max} < \boldsymbol{\delta} < \delta_{max} \tag{28}$$

5) Compute the tightened constraints for state space and controls as:

$$\mathbb{X} \ominus Z \tag{29}$$

$$\mathbb{U} \ominus KZ \tag{30}$$

Use Eq. 29 and Eq. 30 as the constraints for MPC as in Eq. 11 and Eq. 12.

IV. EXPERIMENTAL RESULTS

In this section, we will present the experimental results we have on the CARLA [19] simulator. The vehicle used in the simulation is a Ford Mustang. We have set up experiments in multiple scenarios including intersections, curves, overtaking, and wrong-way driving. The reference velocity for all test cases is set to 20 m/s (roughly 45mph). The disturbance is bounded: $\boldsymbol{w} \in W = \{\boldsymbol{w} \mid \|\boldsymbol{w}\| \leq 0.075\}$. We also compare the performance and safety of our framework and the existing Safe Planning framework in similar settings.

The ego-vehicle is pictured in blue with its transparency decreased over time (i.e. white is the frame at t_0, the less transparent the color is the later a frame is in time). Target vehicles are depicted in red and the transparency works the same way. The Cartesian bounds from IRIS are pictured as polygons in green. The finite planning horizon for CILQR is set to 3 seconds with a discretization of 0.1 seconds. For MPC, the planning horizon is 0.25 seconds with a discretization of 0.05 seconds.

We present the following three scenarios below:

1) The ego-vehicle is travelling behind a slow-moving target vehicle. It safely overtakes the target vehicle in the presence of other vehicles in the adjacent lanes.
2) The ego-vehicle is travelling through an uncontrolled intersection. It safely maneuvers around a target vehicle moving in a direction perpendicular to its motion.
3) The ego-vehicle overtakes a target vehicle on a curved road.

The video demonstrations of the scenarios are available at: `sites.google.com/view/shiveshkhaitan-riss-2020`. Following are the descriptions and results from the above scenarios:

1) Overtake: In this scenario, a target vehicle is travelling in front of the ego-vehicle at 8m/s. The ego-vehicle starts from rest and accelerates while approaching the target vehicle to achieve the target velocity of 20m/s. As seen in Figure 3, the ego-vehicle first decelerates to keep safe-distance from the target vehicle and begins the lane change. After passing the target vehicle it is faced by another parked vehicle. The ego-vehicle then immediately steers back to the original lane keeping a safe distance from the vehicle behind and finally achieves its reference velocity 20 m/s again.

The results for the comparison are listed in Tab. I. It can be seen that the average velocity and time to complete the maneuver are nearly the same for both the frameworks. But, the average distance from the obstacles is much higher with the robust tube MPC. Thus, it can be concluded that that the robust tube controller is safer and guarantees safety without making the controls too conservative.

TABLE I: Comparison in overtaking scenario

Method	CILQR	Robust Tube MPC
Time to complete(s)	17.02	17.31
Average velocity(m/s)	13.20	12.99
Average distance from nearby obstacles(m)	0.18	0.25
Minimum distance from obstacles(m)	0.13	0.13

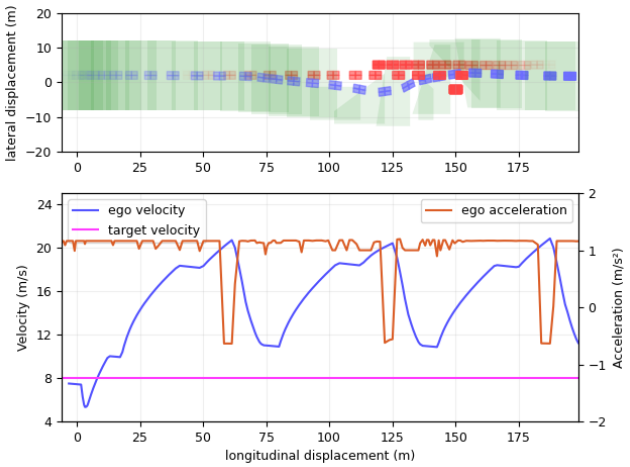


Fig. 3: Overtaking in the presence of obstacles in adjacent lane

2) Intersection: In this scenario, a target vehicle is travelling perpendicular to the ego-vehicle in an uncontrolled intersection at 8m/s. The ego-vehicle starts from rest and is accelerating while approaching the intersection to achieve the reference velocity of 20m/s. As seen in Figure 4, at around a perpendicular distance of 10 meters from the target vehicle, the ego-vehicle starts to steer to avoid the target vehicle. It drops the acceleration initially while turning but starts increasing the acceleration again when it just nears the target vehicle. After avoiding the vehicle, it steers towards the reference lane and finally achieves its reference velocity 20 m/s again. The reference velocity of the ego-vehicle dictates the controller's behaviour regarding slowing down and letting the target vehicle to go or steering to cross the intersection first. In this case, the target velocity is high, thus the ego-vehicle steers and goes first.

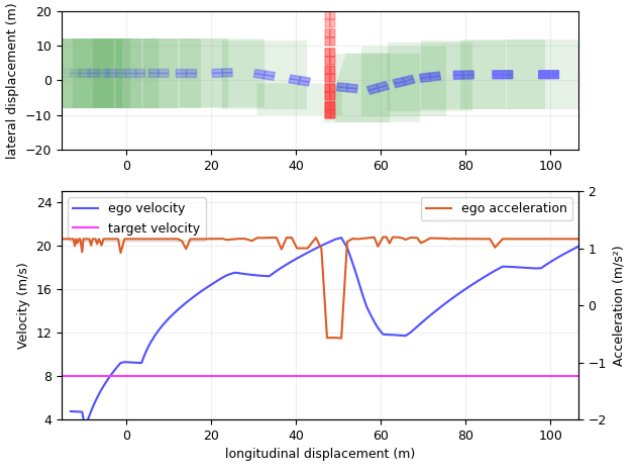


Fig. 4: Obstacle avoidance in intersection

The results for the comparison are listed in Tab. II. The results are similar to the overtaking scenario and validates them.

TABLE II: Comparison in intersection scenario

Method	CILQR	Robust Tube MPC
Time to complete(s)	16.02	16.50
Average velocity(m/s)	14.04	13.63
Average distance from nearby obstacles(m)	0.22	0.31
Minimum distance from obstacles(m)	0.10	0.12

3) Curved road: All scenarios above are tested on straight lanes. In this scenario, a slow vehicle is initially 20 meters ahead of the ego-vehicle on a curved road. The target vehicle is travelling at 6 m/s and the ego-vehicle successfully overtakes the target vehicle in the curve while accelerating to achieve the target velocity of 20 m/s. During the overtake, the ego-vehicle is travelling at 15m/s and then it steers back to its reference lane and accelerates to achieve the reference velocity as seen in the velocity profile of Figure 5.

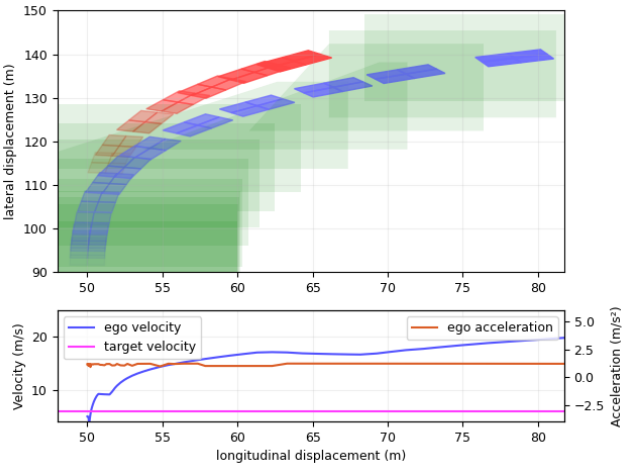


Fig. 5: Overtaking in a curved road

The MPC ensures that in all cases, the ego-vehicle is well within the safe regions. The average loop time for the MPC is 50ms which makes it suitable for real-time implementation. The simulation is written in Python. Gurobi [20] is used for solving the MPC. The control framework runs on a laptop with a 2.50GHz Intel Core i5-7200U CPU.

V. CONCLUSION

In this paper, an end-to-end framework guaranteeing safety across planning and control layers of autonomous vehicles is presented. The method works for non-linear systems and is able to safely avoid dynamic obstacles. The framework is also tested for real-time implementation in the high-fidelity CARLA simulator. This ensures that it is applicable for real-world applications. Experiments also show that it does not make the behaviour much conservative.

Further work in this approach includes optimizing the performance of the low-level controller by making it light-weight. We also hope to reduce the jerks due to extreme acceleration and steering commands.

VI. ACKNOWLEDGMENT

This work was supported by the CMU Robotics Institute Summer Scholars (RISS) Program. Shivesh Khaitan would also like to thank Dr. John M. Dolan and Ms. Rachel Burcin for their constant support.

REFERENCES

[1] Y. Pan, Q. Lin, H. Shah, and J. M. Dolan, "Safe planning for self-driving via adaptive constrained ilqr," in *2020 International Conference on Intelligent RObots and and Systems (IROS)*. IEEE, 2020.

[2] W. Xu, J. Pan, J. Wei, and J. M. Dolan, "Motion planning under uncertainty for on-road autonomous driving," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2507–2512.

[3] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *ASME 2010 dynamic systems and control conference*. American Society of Mechanical Engineers, 2010, pp. 265–272.

[4] A. Arab, K. Yu, J. Yi, and D. Song, "Motion planning for aggressive autonomous vehicle maneuvers," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 221–226.

[5] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2016.

[6] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "Mpc-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2, pp. 265–291, 2005.

[7] J. Löfberg, *Minimax approaches to robust model predictive control*. Linköping University Electronic Press, 2003, vol. 812.

[8] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[9] K. M. M. Rathai, J. Amirthalingam, and B. Jayaraman, "Robust tube-mpc based lane keeping system for autonomous driving vehicles," in *Proceedings of the Advances in Robotics*, 2017, pp. 1–6.

[10] S. S. M. P. Jean-Jacques and E. Slotine, "Tube-based mpc: a contraction theory approach."

[11] B. T. Lopez, J. P. Howl, and J.-J. E. Slotine, "Dynamic tube mpc for nonlinear systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1655–1662.

[12] D. Q. Mayne and E. C. Kerrigan, "TUBE-BASED ROBUST NONLINEAR MODEL PREDICTIVE CONTROL1," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 36–41, 2007.

[13] G. Garimella, M. Sheckells, J. Moore, and M. Kobilarov, "Robust obstacle avoidance using tube NMPC," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, jun 2018.

[14] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *2019 18th European Control Conference (ECC)*, 2019, pp. 811–817.

[15] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Cham: Springer International Publishing, 2015, pp. 109–124. [Online]. Available: https://doi.org/10.1007/978-3-319-16595-0{_}7

[16] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–7.

[17] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative lqr," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 244–254, 2019.

[18] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[20] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

Slip Ratio Estimation for the Localization of Planetary Rovers Using Proprioceptive Sensors

Urara Kono[1], Raewyn Duvall[2] and William L. Whittaker[2]

***Abstract*— This paper proposes a method for estimating the slip ratio of planetary rovers. The estimator is designed based on the equations of motion. Input torque, encoders, and IMU data are used, and driving force, vehicle speed, and slip rate are estimated. This method does not depend on cameras or the soil parameters, and avoids the integral of IMU acceleration with errors. Since the estimator does not use cameras, it will be the back-up for visual odometry (VO) and effective in dark or feature-less places. The sampling time of torque is very short, and that allows us to obtain the slip rate much faster than VO. The simulation in MATLAB & Simulink showed improvements in wheel odometry. As future work, the estimator can be fused with VO and fill the gaps between keyframes.**

***Index Terms*— Localization, Wheeled Robots, Space Robotics and Automation.**

I. INTRODUCTION

Planetary rovers have contributed to discovering new scientific knowledge and are still expected to do so. Localization is important for the rovers to reach the destinations and avoid the obstacles on unknown, nonhomogeneous terrain. Autonomous driving enables a rover to cover longer distance per sol - a Martian solar day - by minimizing communication with operators on Earth [1].

The localization of planetary rovers is different from that on Earth from the following points [2]: first, Global Navigation Satellite System (GNSS) is denied in space, which means no global, ground truth positioning; second, rovers' computational availability is limited compared to current computers since they are designed to be resistant to cosmic rays; third, it takes some time to communicate between the Earth and any other planetary body, making communication costly in time; These points make real-time localization challenging.

Wheel odometry is the fundamental basis for localization. It takes encoder data and computes how far the rover has traversed. This method is simple, takes minimal computation, and is effective in flat and hard terrains. In the Mars Exploration Rover (MER) mission, when the rover ran at the flat ground for 2km, the accumulated error was only 3% [3]. However, on a sloped terrain such as 25°, it slipped at as much as 125% [4]. Something must be done to tackle these errors caused by slip.

Visual odometry (VO) is currently the primary method to minimize this error. It is considered a key technology to realize reliable fully-autonomous navigation. It consists of four processes: feature detection, matching of stereo camera based on the features, feature traction, and robust attitude estimation [4]. VO was first implemented in MER [4]. In NASA's Mars Science Laboratory (MSL) mission, it was used heavily on Curiosity to detect unexpected slip quickly, so that the rover can traverse terrain as fast and far as possible [5].

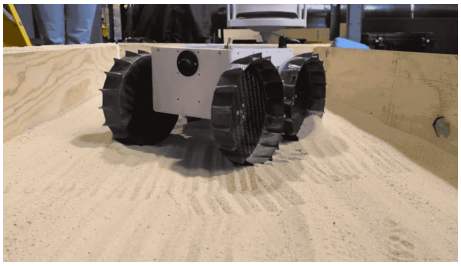


Fig. 1. CMU Iris [6]

Although VO works well, errors still exist as cameras cannot see well in dark places or feature-less terrains - such as those on the Moon or Mars - and can cause feature matching to fail. Terramechanics have worked on slip estimation by making precise models of the ground and the wheel [7], [8]. However, this uses the soil parameters, which are unknown in new environments such as those planetary rovers traverse. Current research tries to estimate the parameters, but that method needs heavy computation [9]. Other research proposes estimating slip using current, but this also uses soil parameters [10].

This paper proposes a slip ratio estimator that does not require cameras or soil parameters. It is model-based and derives the equations from the motion of the rover. The driving force is estimated from the disturbance observer [11], and the rover speed is calculated with the angle speed from the encoders and the input torque. This method is robust to the noise because it does not integrate the acceleration from the IMU with offsets. It was first developed on electric vehicles (EVs) with In Wheel Motors (IWM) by [12], [13] and used for the driving force control (DFC) of EVs [14], [15]. Since no cameras or feature extractions are needed, it can work in dark places like moon poles and holes or feature-less terrains. This method is much less computationally demanding than VO and will be useful for rovers with limited processing. Since the torque response of an electric motor is short - a few milliseconds, the slip ratio can be obtained at higher rates than VO. Therefore, this method will

[1] Urara Kono is with faculty of Electrical and Electronic Engineering, The University of Tokyo, Tokyo, Japan. urara27.k@gmail.com

[2]Raewyn Duvall and William L. Whittaker are with Robotics Institute, Carnegie Mellon University, Pittsburgh, USA. {rduvall, ww0t}@andrew.cmu.edu

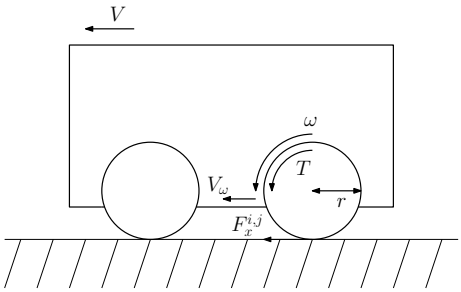


Fig. 2. Rover model

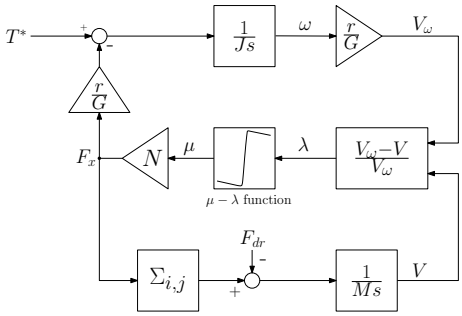


Fig. 3. The block diagram of the plant

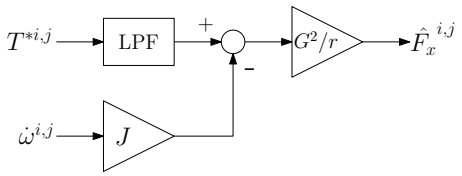


Fig. 4. The diagram of Driving Force Observer (DFO)

improve the localization accuracy while the driving between keyframes or waypoints.

This research is focused on the use case for The Carnegie Mellon University (CMU) Iris (Fig. 1). It is a micro rover with very limited computation power and memory. Low resolution images and other sensor data will be sent to Earth at a data rate of 50kbps and with a minimal delay of 8 seconds; the state is then estimated on the ground. Future micro-rovers, like the joint development by CMU and Astrobotic of CubeRover and Rover Teams, will similarly have limited onboard processing. MoonRanger is another joint rover that will be flying to the lunar pole in 2022. It has a GPU that handles image processing almost in real-time. However, if the GPU fails, MoonRanger will have to navigate back to the lander on a more limited backup processor, with no visual input. Every piece of additional information on the positioning or state estimation derived with minimal computation is beneficial.

The remainder of this paper is organized as follows: In Section 2, the rover model and slip estimation method are described. The simulation results are presented in Section 3. Section 4 is discussion, and Section 5 gives the conclusions.

II. MODEL AND METHOD

A. Rover model

The longitudinal motion equations of the wheels and the body model, shown in Fig. 2, are described as

$$J\dot{\omega}^{ij} = T^{ij} - (r/G)F_x^{ij}, \tag{1}$$

$$M\dot{V} = \Sigma F_x^{ij} - F_{dr}, \tag{2}$$

$$V_\omega^{ij} = (r/G)\omega^{ij}, \tag{3}$$

where J is the wheel inertia, ω is the wheel speed, T is the motor torque, r is the radius of the wheels, G is the reduction ratio, F_x^{ij} is the driving force (i = front, rear, j = left, right), F_{dr} is the running resistance, M is the vehicle mass, V is the rover speed, and V_ω is the wheel speed. The slip ratio is defined as

$$\lambda^{ij} := \frac{V_\omega^{ij} - V}{\max(V_\omega^{ij}, V, \epsilon)}, \tag{4}$$

where $\epsilon \ll 0.01$ is the small constant to avoid zero denominator. In this paper, the rover is running straight, and the driving force can be described as

$$\Sigma F_x^{ij} = \mu N, \tag{5}$$

where N is the nominal reaction force. The block diagram of the plant is described in Fig. 3. The driving force F_x^{ij} is estimated using Driving Force Observer (DFO) [11], and its diagram is shown in Fig. 4.

In the Sections II-B and II-C, the slip estimator for driving mode and braking mode is designed.

B. Driving mode

When the rover is driving, $\max(V_\omega, V, \epsilon) = V_\omega$. By differentiating (4) and using (1) and (2), the wheel acceleration is described as

$$\dot{\omega}^{ij} = \frac{T^{ij} - (r/G)F_{dr} + (r/G)^2 M\omega^{ij}\dot{\lambda}^{ij}}{J + (r/G)^2 M(1-\lambda^{ij})}. \tag{6}$$

In order to estimate the slip ratio, this equation is transformed to the differential equation of λ^{ij}:

$$\dot{\lambda}^{ij} = -\frac{\dot{\omega}^{ij}}{\omega^{ij}}\lambda^{ij} + \left(1 + \frac{J}{(r/G)^2 M}\right)\frac{\dot{\omega}^{ij}}{\omega^{ij}} - \frac{T^{ij}}{(r/G)^2 M\omega^{ij}} + \frac{F_{dr}}{(r/G)M\omega^{ij}}. \tag{7}$$

The slip estimator is designed by putting hats on unmeasured terms in (7):

$$\dot{\hat{\lambda}}^{ij} = -\frac{\dot{\omega}^{ij}}{\omega^{ij}}\hat{\lambda}^{ij} + \left(1 + \frac{J}{(r/G)^2 M}\right)\frac{\dot{\omega}^{ij}}{\omega^{ij}} - \frac{T^{ij}}{(r/G)^2 M\omega^{ij}} + \frac{\hat{F}_{dr}}{(r/G)M\omega^{ij}}. \tag{8}$$

$\hat{F}_{dr}$, with modeling errors or disturbance, can be calculated from (2) using $\dot{V}$ from the IMU and F_x^{ij} from the DFO.

When $\hat{\lambda}^{ij}$ converges, the estimation error will also converge, defined as

$$e^{ij} = \lambda^{ij} - \hat{\lambda}^{ij}. \tag{9}$$

By subtracing (8) from (7), this differential equation of e^{ij} is obtained:

$$\dot{e}^{ij} = -\frac{\dot{\omega}^{ij}}{\omega^{ij}}e + \frac{F_{dr} - \hat{F}_{dr}}{(r/G)M\omega^{ij}}. \tag{10}$$

If $F_{dr} = \hat{F}_{dr}$ and $\frac{\dot{\omega}^{ij}}{\omega^{ij}} > 0$, the estimation error will converge to zero.

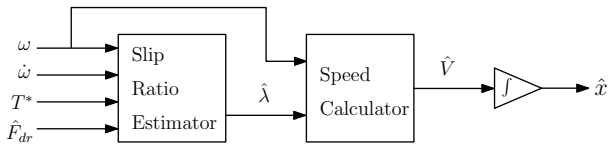


Fig. 5. The block diagram of the whole system

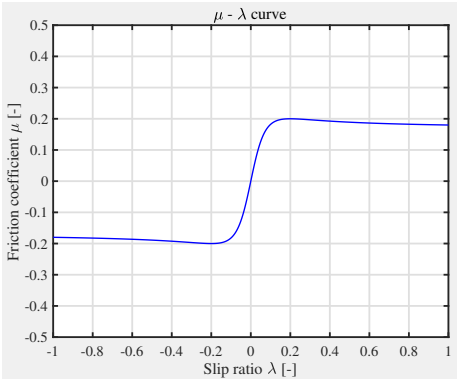


Fig. 6. The $\mu - \lambda$ curve

C. Braking mode

When the rover is braking, $\max(V_\omega, V, \epsilon) = V$. By differentiating (4) and using (1) and (2), Equation (11) is obtained:

$$\dot{\lambda}^{ij} = \frac{\dot{\omega}^{ij}}{\omega^{ij}}\left(1+\lambda^{ij}\right) - \left(\frac{T^{ij} - J\dot{\omega}^{ij} - (r/G)F_{dr}}{r^2 M\omega^{ij}}\right)\left(1+\lambda^{ij}\right)^2. \tag{11}$$

Based on (11), the slip estimator is defined as

$$\dot{\hat{\lambda}}^{ij} = \frac{\dot{\omega}^{ij}}{\omega^{ij}}\left(1+\hat{\lambda}^{ij}\right) - \left(\frac{T^{ij} - J\dot{\omega}^{ij} - (r/G)\hat{F}_{dr}}{(r/G)^2 M\omega^{ij}}\right)\left(1+\hat{\lambda}^{ij}\right)^2. \tag{12}$$

By subtracing (12) from (11), this differential equation is obtained:

$$\dot{e}^{ij} = \left\{\frac{\dot{\omega}^{ij}}{\omega^{ij}} - \left(\frac{T^{ij} - J\dot{\omega}^{ij} - (r/G)\hat{F}_{dr}}{r^2 M\omega^{ij}}\right)(\lambda^{ij} + \hat{\lambda}^{ij} + 2)\right\} e^{ij}. \tag{13}$$

$\hat{F}_{dr}$ will converge to the true value with the time constant for DFO, and $F_{dr} \simeq \hat{F}_{dr}$ is assumed. Using $\dot{V}_\omega^{ij}, V_\omega^{ij}$, and $\dot{V}$, (13) can also be written as

$$\dot{e}^{ij} = \left(\frac{\dot{V}_\omega^{ij}}{V_\omega^{ij}} - \frac{\dot{V}}{V_\omega^{ij}}(2 + \lambda^{ij} + \hat{\lambda}^{ij})\right). \tag{14}$$

The error e^{ij} will converge if $\dot{V}_\omega^{ij} - \dot{V}^{ij}(2 + \lambda^{ij} + \hat{\lambda}^{ij}) < 0$. The block diagram of the whole localization system is represented in Fig. 5.

III. SIMULATION

The simulation was conducted on MATLAB & Simulink. In (5), magic formula [16] was used on the simulation and is described as

$$\mu(\lambda) = D\sin\left(C\tan^{-1}\left(B(1-E)\lambda + E\tan^{-1}(B\lambda)\right)\right). \tag{15}$$

The hight of the curve and the λ value where the curve gets its peak was both set 0.2 based on [17], which conducted

TABLE I

PARAMETERS OF TERRAMECHANICS OF SAND [17]

Terrain parameter	Value
n	1.10
$k_c\left(\mathrm{kPa/m}^{n-1}\right)$	0.95
$k_\varphi\,(\mathrm{kPa/m}^n)$	1528.43
c(kPa)	1.04
$\varphi\,(^\circ)$	28
K(m)	0.0254

TABLE II

ROVER PARAMETERS OF IRIS

Rover parameter	Value
r(cm)	8.92
$J(\mathrm{kgm}^2)$	2.20×10^{-7}
G	794
M(kg)	2.38

rover experiments with sand as depicted in Table I, where the parameters are soil deformation index n, cohesive and frictional moduli of deformation k_c and k_φ, soil cohesion coefficient c, soil friction angle φ, and soil shear modulus of deformation K. The graph of this curve is shown on Fig. 6. It was used to produce true speed and position data in the simulation, not in the estimation process.

In the simulation, IMU data was not used. We assumed that the rover is running on flat terrain, and the driving resistance is only the rolling resistance. Since there is no atmosphere on the moon, air resistance was neglected. The estimated running resistance is described as

$$\hat{F}_{dr} = \mu_{\mathrm{roll}} N, \tag{16}$$

where μ_{roll} is the rolling resistance coeficient, and its value is 0.02. It is based on the data from the Lunar Roving Vehicle in Apollo mission [18].

The simulation was conducted based on Iris rover model; its physical parameters are presented in Table II.

IV. SIMULATION RESULTS

Two different step input torques were given at t = 0.1s to show the feasibility of the slip estimator.

First, the input torque was $T = 1.9\times10^{-4}\mathrm{N}\cdot\mathrm{m}$, which was actually given at a flat terrain test when the rover ran straight. The graph of the rover speed is shown in Fig. 7. Without slip estimation, there was no slip compensation for the encoders data. The rover hardly slipped ($\lambda = 0.03$), and there was no significant difference in the position either as shown in Fig. 8.

Second, the input torque was $T = 9.4\times10^{-4}\mathrm{N}\cdot\mathrm{m}$, which is half the stall torque. The rover slipped a lot, and its slip ratio was ($\lambda = 0.41$). The graphs of the rover speed and position are shown in Fig. 9 and Fig. 10, respectively. Significant difference was observed between with and without slip estimation.

Single-step input does not represent the actual input, which is complex due to PI control so that the speed of the motor can be constant. In the next simulation, the input included both positive and negative, which was extracted from the

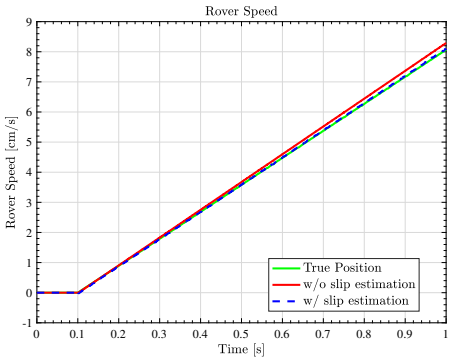


Fig. 7. Rover Speed ($T = 1.9 \times 10^{-4}$N · m)

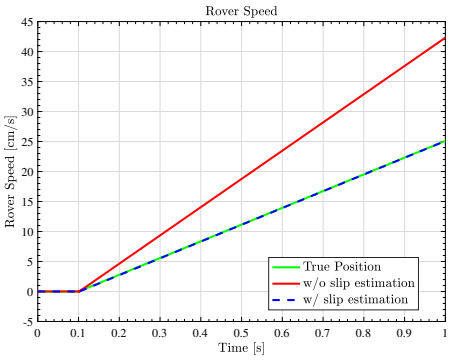


Fig. 9. Rover Speed ($T = 9.4 \times 10^{-4}$N · m)

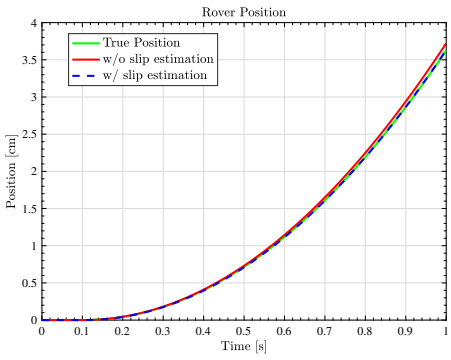


Fig. 8. Rover Position ($T = 1.9 \times 10^{-4}$N · m)

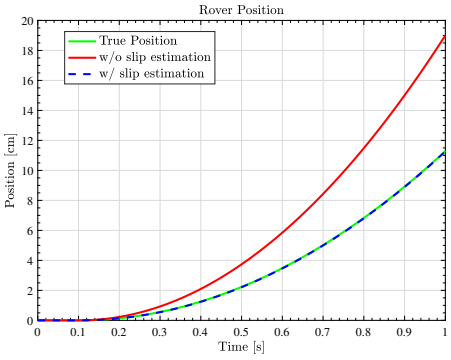


Fig. 10. Rover Position ($T = 9.4 \times 10^{-4}$N · m)

actual input given at a flat-field test, when the rover was running straight. The input torque is shown in Fig. 11. The speed and the position graphs are Fig. 12 and Fig. 13, respectively. The slip estimator worked even when the input torque changed between positive and negative.

V. DISCUSSION

In this section, the fidelity of the simulation is discussed. As shown in Fig. 1, Iris has grousers on its wheels, which will prevent slip to some extent. However, the equations (1) and (2) do not consider the effect of the driving force which work on the grousers. Since the proposed method is not physically tested due to remote research, their effect is unknown. Furthermore, since Iris is very small compared to most rovers, it might be more easily affected by uneven terrains, which might cause forces from unexpected directions. In the simulation, simple equations were used that did not factor these additional sources of error. In future modeling, the errors and disturbances should be taken into account. For example, the following equations based on Fig. 14 have error terms N_m $(m = 1, 2, 3, 4)$.

$$m\dot{V} = \Sigma F_x^{ij} + N_1, \tag{17}$$

$$J\dot{\omega}^{ij} = T^{ij} - rF_x^{ij} + N_2, \tag{18}$$

$$mV(\frac{d\beta}{dt} + \gamma) = \Sigma F_y^{ij} + N_3, \tag{19}$$

$$I\frac{d\gamma}{dt} = 2l_f F_y^{f,j} - 2l_r F_y^{r,j} + N_4, \tag{20}$$

where F_y^{ij} is the driving force which acts vertical to the direction of the wheel. In [19], only yaw moment disturbance N_3 was considered and equations (19) and (20) were used for the trajectory tracking and the stabilization of agricultural vehicles. These error terms will not be observable due to limited kinds of equipped sensors on rovers, but they can be estimated using disturbance observer or appropriate approximations. N_1 is the same with F_{dr} and can be calculated using $\dot{V}$ from the IMU and F_x^{ij} from the DFO. $N_2 \ll F_x^{ij}$ and $N_2 \simeq 0$ can be approximated. Likewise, N_3 can be calculated, and $N_4 \simeq 0$ can be approximated.

VI. CONCLUSIONS

VO has been used to improve the accuracy of localization by wheel odometry and IMU. However, it is still important to find methods that do not use cameras for these reasons:

- Those methods can reduce the dependence on VO for rovers with limited computation or as a backup.
- VO sometimes cannot work well on non-feature terrain.
- Cameras cannot see anything at dark places like the moon poles or holes.
- VO is essentially blind driving between keyframes or waypoints.

In order to work around the limitation of or improve VO, this paper proposed the slip ratio estimator using proprioceptive sensors. It is model-based and does not use cameras or soil parameters. It meets these demands mentioned above and compensates for the wheel odometry error caused by slip

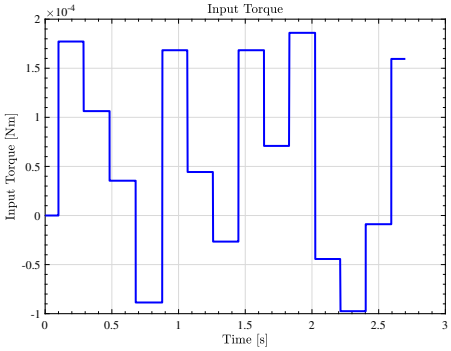


Fig. 11. Input torque

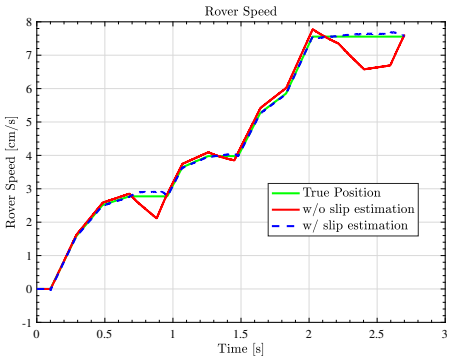


Fig. 12. Rover Speed

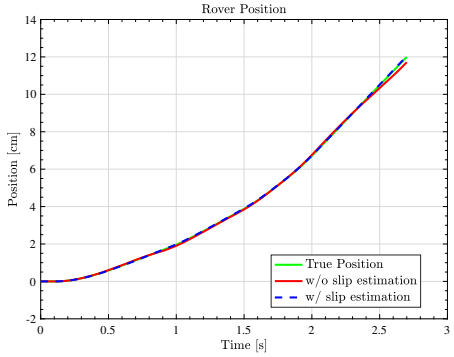


Fig. 13. Rover Position

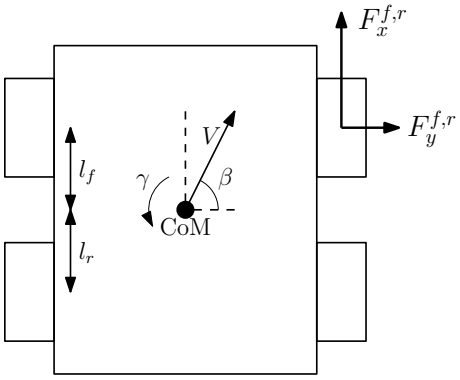


Fig. 14. Rover Model

on the simulation. Experiments could not be conducted due to remote research.

In the future, if rovers run much faster, the pictures could be blurred [20] and non-VO localization will be expected to play a more important role in localization. The sampling time of proprioceptive sensors and torque input is short and a small amount of calculation is needed, so the slip estimator can make up for the gap between keyframes.

However, the proposed method has some issues that need to be improved: it may not model the rover well and modeling errors need to be considered, as mentioned in the Discussion section. Since little research has worked on the improvement of localization using proprioceptive sensors, it is important to find the limitation of the estimator with experiments on inclined or uneven terrain. When the estimator is proven to be effective, fusing the slip estimator with Visual Inertial Odometry would minimize current errors in planetary rovers' localization.

ACKNOWLEDGMENT

This work presented here received funding from CMU RISS Program Scholarship and CMU RI Lab Scholarship. The authors would like to thank all members of Iris project, Ms. Rachel Burcin, Prof. John Dolan, RISS alumni and all the sponsors for making this wonderful opportunity in the midst of uncertainty.

REFERENCES

[1] K. Otsu, A. A. Agha-Mohammadi, and M. Paton, "Where to Look? Predictive Perception with Applications to Planetary Exploration," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 635–642, 2018.

[2] J. Strader, K. Otsu, and A. akbar Agha-mohammadi, "Perception-aware autonomous mast motion planning for planetary exploration rovers," *Journal of Field Robotics*, no. December 2018, 2019.

[3] J. J. Biesiadecki, P. C. Leger, and M. W. Maimone, "Tradeoffs between directed and autonomous driving on the Mars exploration rovers," *International Journal of Robotics Research*, vol. 26, no. 1, pp. 91–104, 2007.

[4] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[5] A. Rankin, M. Maimone, J. Biesiadecki, N. Patel, D. Levine, and O. Toupet, "Driving Curiosity: Mars Rover Mobility Trends During the First Seven Years," no. Lm, 2019.

[6] Techcrunch, "CMU's tiny robot rover passes NASA design review ahead of 2021 trip to the Moon." [Online]. Available: https://techcrunch.com/2020/05/12/cmus-tiny-robot-rover-passes-nasa-design-review-ahead-of-2021-trip-to-the-moon/

[7] L. Ding, H. Gao, Z. Deng, K. Yoshida, and K. Nagatani, "Slip ratio for lugged wheel of planetary rover in deformable soil: Definition and estimation," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 3343–3348, 2009.

[8] K. Yoshida and H. Hamano, "Motion dynamics of a rover with slip-based traction model," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, no. May, 2002, pp. 3155–3160.

[9] Shamrao, C. Padmanabhan, S. Gupta, and A. Mylswamy, "Estimation of terramechanics parameters of wheel-soil interaction model using particle filtering," *Journal of Terramechanics*, vol. 79, pp. 79–95, 2018. [Online]. Available: https://doi.org/10.1016/j.jterra.2018.07.003

[10] L. Ojeda, D. Cruz, G. Reina, and J. Borenstein, "Current-based slippage detection and odometry correction for mobile robots and

planetary rovers," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 366–378, 2006.

[11] Y. Hori, "Future vehicle driven by electricity and control - Research on four-wheel-motored "UOT Electric March II"," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 5, pp. 954–962, 2004.

[12] K. Fujii and H. Fujimoto, "Traction control based on slip ratio estimation without detecting vehicle speed for electric vehicle," in *Fourth Power Conversion Conference-NAGOYA, PCC-NAGOYA 2007 - Conference Proceedings*, 2007.

[13] T. Suzuki and H. Fujimoto, "Slip ratio estimation and regenerative brake control without detection of vehicle velocity and acceleration for electric vehicle at urgent brake-turning," in *2010 11th IEEE International Workshop on Advanced Motion Control (AMC)*. IEEE, mar 2010, pp. 273–278. [Online]. Available: http://ieeexplore.ieee.org/document/5464122/

[14] H. Fujimoto, K. Fujii, and N. Takahashi, "Vehicle stability control of electric vehicle with slip-ratio and cornering stiffness estimation," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, no. 1, pp. 2–7, 2007.

[15] K. Maeda, H. Fujimoto, and Y. Hori, "Four-wheel driving-force distribution method based on driving stiffness and slip ratio estimation for electric vehicle with in-wheel motors," in *2012 IEEE Vehicle Power and Propulsion Conference*. IEEE, oct 2012, pp. 1286–1291. [Online]. Available: http://ieeexplore.ieee.org/document/6422490/

[16] H. B. PACEJKA and I. J. M. BESSELINK, "Magic Formula Tyre Model with Transient Properties," *Vehicle System Dynamics*, vol. 27, no. sup001, pp. 234–249, 1997. [Online]. Available: https://doi.org/10.1080/00423119708969658

[17] Z. Li and Y. Wang, "Coordinated control of slip ratio for wheeled mobile robots climbing loose sloped terrain," *Scientific World Journal*, vol. 2014, 2014.

[18] H. Furuhata, "Progress of a manned, pressurized rover- First consideration and future work- (Japanese)," Tech. Rep., 2020. [Online]. Available: http://www.ihub-tansa.jaxa.jp/files/20200207/20200207_furuhata.pdf

[19] M. Suzuki, H. Fujimoto, and Y. Hori, "The Simultaneous Estimation Method of Terrain Parameter and Vehicle Dynamics Variables for Agricultural Vehicle," *Proceedings - 2019 IEEE International Conference on Mechatronics, ICM 2019*, no. 1, pp. 596–601, 2019.

[20] R. Gonzalez and K. Iagnemma, "Slippage estimation and compensation for planetary exploration rovers. State of the art and future challenges," *Journal of Field Robotics*, vol. 35, no. 4, pp. 564–577, 2018.

Localization for Autonomous Polar Lunar Micro Roving

Oleksandr Koreiba[1], Haidar Jamal[2], Heather Jones[2] and William L."Red" Whittaker[2]

***Abstract*— Autonomous traversing of the micro rovers poses serious challenges due to limiting computational capabilities as well as the uncontrolled and harsh environments. Rovers such as MoonRanger will need to be capable of performing long-range scientific missions outside of the lander's communication zone hence need to accurately localize themselves to allow autonomous navigation. This research investigates implementing an extended Kalman filter for accurate rover orientation estimation and localization. The developed method consists of two parts - attitude and heading reference system that calculates the rover's orientation and position estimation algorithm, that combines the computed orientation with the data from wheel encoders to determine the position and heading of the rover. Testing has shown that the algorithm outperforms the dead-reckoning method of position estimation and provides accurate state estimation.**

***Index Terms*— Space robotics and automation, Localization, Wheeled Robots**

I. INTRODUCTION

Recent initiatives for escalating the rate of interplanetary exploration require a lot of research and analysis to be done in advance. Essential data can be gathered by micro rovers [1], as they are easier and cheaper to deploy and take less time to develop. Most of the deployed rovers have limited autonomy, their actions were planned in advance and controlled by humans [2] [3]. Smaller rovers, like MoonRanger (Fig. 1) will need to possess a higher degree of autonomy as they do not have the power and space capabilities of carrying hardware to communicate directly with Earth. Since large portions of long traversing missions will require robots to traverse beyond the lander communication range, micro rovers need to have exceptional navigation to be able to return to the ground base. Accurate localization of the rover on both short and long traversing mission make the rover's ability to correctly estimate its position and heading essential. Highly accurate data can be obtained from gyroscopes and accelerometers in the inertial measurement unit (IMU), as well as wheel encoders. However, they are prone to drift and noise, gradually accumulating error and contributing to a faulty state estimation.

One of the simplest solutions to the localization problem is using a dead-reckoning algorithm that estimates the position of the rover using the data from wheel encoders. Unfortunately, relying solely on such estimation is impossible as it drifts a lot over time. For more a robust localization sensor fusion needs to be leveraged [4] [5]. This research looks into developing a state estimation algorithm based on the extended Kalman filter [6] that consists of two parts - Attitude and heading reference system (AHRS), and position estimation. The algorithm is tested on PowerRanger, which is a development rover for future MoonRanger mission designed to autonomously traverse the south pole of the Moon in search of ice.

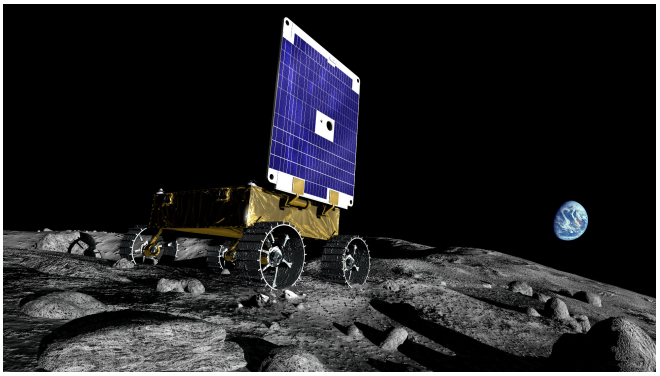


Fig. 1. MoonRanger rover

Section 2 of this paper presents the structure and development of the AHRS and position estimation algorithms. Section 3 describes tests, performed on a lunar-like terrain with PowerRanger development rover and results. Finally, section 4 draws conclusions and talks about the possibilities of future work and the future improvements to the algorithm.

II. METHODS

A. Extended Kalman filter

EKF is a commonly used filter for non-linear state estimation problems. The algorithm is recursive and consists of two steps - prediction, based on the predetermined system model and measurement update, using the input from the sensors. The system model in the prediction step shows how the state x_t changes with time in the presence of noise:

$$x_t = g(u_t, x_{t-1} + \epsilon_t) \tag{1}$$

where g is a non-linear function, u_t is the control input and ϵ_t is white Gaussian noise. The measurement model z_t relates data received by sensors to the state:

$$z_t = h(x_t + \delta_t) \tag{2}$$

where h is a non-linear function and δ_t white Gaussian noise.

The beliefs about the rover's state are represented as the first two moments (mean and covariance) of a multivariate

[1]Oleksandr Koreiba is with Faculty of Electrical and Electronics Engineering, Department of Automation, Kaunas University of Technology, Kaunas, Lithuania oleksandr.koreiba@ktu.edu,

[2]Haidar Jamal, Heather Jones and William L. "Red" Whittaker are with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, USA hjamal@andrew.cmu.edu, hjones@andrew.cmu.edu, red.cmu.edu

Gaussian distribution. Prediction step approximates the new state x_{t+1} as well as covariance matrix (P_{t+1}, t) associated with it. When the measurement is received, the Kalman gain is calculated, which defines the relative importance of the measurement versus state estimate. If the error in measurement is high the gain is lower, and the systems pays more attention to the prediction; if the gain is high-measurements are trusted more. After this, the covariance is updated and the previous state is assigned to the new state in the next iteration of the algorithm.

B. AHRS

An attitude and heading reference system is a combination of instruments capable of maintaining an accurate estimate of the vehicle's roll ϕ, pitch θ, and yaw ψ as the vehicle maneuvers [7]. The development Inertial Measurement Unit has an accelerometer, a gyroscope, and a magnetometer, however, only data from the first two will be considered, as flight IMU of the MoonRanger will not have magnetometer. Data gathered from the sensors is highly accurate over the short period of time, however, gyroscope readings drift over time, and accelerometer is susceptible to noise, which makes it unreliable over time. AHRS algorithm fuses measurements from sensors using an extended Kalman Filter, which significantly reduces drift and improves the estimate. State vector is represented as following $x = [b, x_g, x_a]$, where b is orientation quaternion, x_g and x_a are gyroscope and accelerometer biases respectively. Before the first iteration of the algorithm, the initial orientation b_0 and gyroscope bias x_{g0} need to be determined. Gravity vector is obtained by averaging the values of accelerometer y_a over a certain period of time T

$$\bar{g}^b = \frac{1}{T}\int_0^T -y_a(\tau)\ d\tau \rightarrow [\bar{g}_1, \bar{g}_2, \bar{g}_3] \tag{3}$$

Roll and pitch are then calculated using the following equations:

$$\theta(T) = \text{atan2}(\bar{g}_2, \bar{g}_3) \tag{4}$$

$$\psi(T) = \text{atan2}(-\bar{g}_1, \sqrt{\bar{g}_2^2, \bar{g}_3^2}) \tag{5}$$

Initial yaw is set to zero and the orientation quaternion b is constructed from the Euler angles values. After initialization is complete, the prediction step of the EKF is executed, where the state vector and the covariance matrix is being approximated using the data from the gyroscope. The predicted estimate of the orientation is obtained from integrating the equation:

$$\hat{b} = \frac{1}{2}\begin{bmatrix} -\hat{b_2} & -\hat{b_3} & -\hat{b_4} \\ \hat{b_1} & \hat{b_4} & -\hat{b_3} \\ -\hat{b_4} & \hat{b_1} & \hat{b_2} \\ \hat{b_3} & -\hat{b_2} & -\hat{b_1} \end{bmatrix} \hat{\omega}^b_{bn} \tag{6}$$

Body angular rate $\hat{\omega}^b_{bn}$ can be obtained by subtracting gyroscope bias $\hat{x_g}$ from it's output u:

$$-\hat{\omega}^b_{bn} = u - \hat{x_g} \tag{7}$$

Covariance matrix is being updated at the end as follows:

$$\bar{P_t} = \Phi_{t-1} P^+_{t-1} \Phi^T_{t-1} + Qd_{t-1} \tag{8}$$

where Φ is a state transition matrix and Qd is a discrete-time process noise matrix. Measurement update is only executed when the robot is stationary, and it is possible to compare the known gravity vector with the measured values. Kalman gain is computed using the formula:

$$K_t = \bar{P_t} H^T_t (R_t + H_t \bar{P_t} H^T_t)^{-1} \tag{9}$$

where H is an accelerometer measurement matrix and R is the measurement covariance. The new state is obtained, then both the covariance matrix and biases are updated.

C. Position estimation

MoonRanger is a four-wheeled skid steer rover, however, in this research, the kinematic model was simplified to the differential drive (Fig. 2) with the assumption that the angular velocities of wheels on each side are the same. The state of the rover is represented by $X = [x, y, \theta]$, where x, y are the position and θ is the heading of the rover. The predicted state is calculated using the kinematic model and ticks from wheel encoders. The distance travelled by each wheel is computed by the formulas:

$$D_l = 2\pi R \frac{\Delta ticks_l}{N} \qquad D_r = 2\pi R \frac{\Delta ticks_r}{N} \tag{10}$$

where R is the wheel radius, $\Delta ticks_l$ and $\Delta ticks_r$ are the number of ticks received from encoders in the sample time period and N is the number of ticks per meter. The distance travelled by the center of the robot is then:

$$D_c = \frac{D_r - D_l}{2} \tag{11}$$

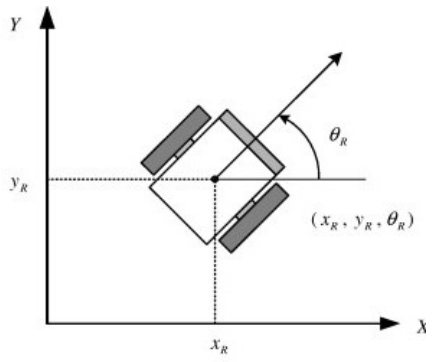


Fig. 2. Differential drive model

Knowing those, the predicted state of the rover is obtained:

$$x' = x + D_c cos\theta \tag{12}$$

$$y' = y + D_c sin\theta \tag{13}$$

$$\theta' = \theta + \frac{D_r - D_l}{L} \tag{14}$$

where L is the width of the rover's base. The covariance matrix is updated using the following formula:

$$P_{t+1} = F_x P_t F^T_x + F_u U_t F^T_u \tag{15}$$

where F_x and F_u are Jacobian matrices with respect to the state and control input respectively, and U is the noise matrix. After the predicted state is calculated, the measurement update is executed. The Kalman gain is calculated as follows:

$$K = \frac{P_t H_t^T}{H_t P_t H_t^T + Q} \tag{16}$$

where H is the measurement Jacobian and Q is measurement noise. The estimated state of the rover, represented as $(x, y, z, \phi, \theta, \psi)$ is obtained and the covariance matrix is updated.

III. RESULTS

A. AHRS

AHRS outputs the orientation of the rover with respect to the world frame. Visualisation of the output was done using the RViz tool as shown in Fig. 3. The robot frame is transformed to the world frame and is published every time a prediction step is executed.

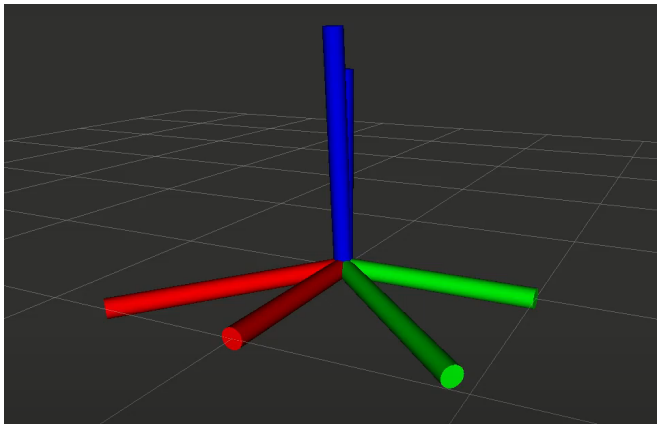


Fig. 3. AHRS output

Evaluation of the position estimation was performed using prerecorded data sets from the PowerRanger rover (Fig. 4). The testing was performed on a lunar-like terrain that simulates traversing on the Moon. The algorithms outputs estimate in the form of x, y, z coordinates relative to the world frame and orientation quaternion that is obtained from the AHRS and updated using the state approximated during the prediction step with wheel encoders. The algorithm generated a more accurate state estimation results when compared to a simple dead reckoning scheme.

IV. CONCLUSIONS

A. Summary

This research looked at developing a localization algorithm based on an extended Kalman filter. Developed state estimation algorithm consists of two parts - AHRS, which fuses data from accelerometer and gyroscope to determine the orientation and position estimation that calculates the state using the data from wheel encoders and orientation from AHRS. The developed method provides superior state estimation when compared to a dead reckoning algorithm.

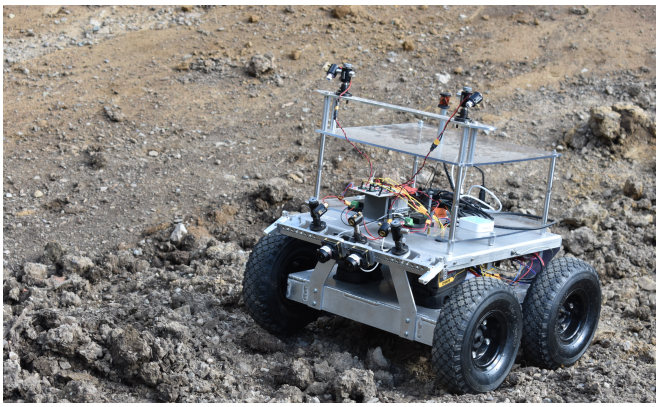


Fig. 4. PowerRanger development rover

B. Future work

Future work will focus on implementing an integrated model, that takes into account accelerometer readings in the position estimation algorithm. To improve the AHRS, data from sun-sensor can be fused to correct for the small drift of yaw over time. The sun-sensor gives the global orientation reference and could be used periodically to update the orientation.

Considering that angular velocities of each wheel will be different due to slippage on loose soil of the Moon, a new kinematic model of the rover should be developed. MoonRanger has encoders on each of the four motors and switching to a four-wheeled skid steer kinematic model will greatly improve the accuracy of the position estimation.

ACKNOWLEDGMENT

I would like to express gratitude to Haidar Jamal, Heather Jones, and William “Red” Whittaker for the guidance during the research as well as members of the MoonRanger team for their help and shared knowledge. I also want to express appreciation to the organisers of RISS program and the sponsors, without whom this experience would not be possible.

REFERENCES

[1] C. M. Angle and R. A. Brooks, “Small planetary rovers,” 1990.
[2] J. C. M. A. E. Mishkin and T. T. Nguyen, “Experiences with operations and autonomy of the mars pathfinder microrover,” *1998 IEEE Aerospace Conference Proceedings (Cat. No.98TH8339)*, vol. 2, pp. 337–351 vol.2, 1998.
[3] D. M. Lyons and J. H. Allton, “Achieving a balance between autonomy and tele-operation in specifying plans for a planetary rover,” in *Other Conferences*, 1991.
[4] P. S. S. e. Eric T. Baumgartner, Patrick C. Leger, “Sensor-fused navigation and manipulation from a planetary rover,” in *Other Conferences*, 1998.
[5] E. T. Baumgartner, H. Aghazarian, and A. Trebi-Ollennu, “Rover localization results for the fido rover,” in *SPIE Optics East*, 2001.
[6] S. I. Roumeliotis and G. A. Bekey, “Extended kalman filter for frequent local and infrequent global sensor data fusion,” in *Other Conferences*, 1997.
[7] J. A.Farrell, *Aiden navigation: GPS with High Rate Sensors*, 1st ed. McGraw-Hill Education, Apr. 2008.

Multi-Agent Path Finding Maximizing Inter-Agent Distance

Sahana Kumar[1], Ishani Chatterjee[2], Tushar Kusnur[2], and Maxim Likhachev[3]

***Abstract*— The development of multi-agent robotic systems with advanced knowledge not only of their own surroundings, but also of each other, is an area of rising prevalence in today's world. Research in this area is applicable in a variety of fields almost perfectly suited to autonomous robotic intervention, including factories, aerial movement, or even computer games. In the past several years, great strides have been taken in collision-free multi-agent path planning. However, there has not yet been an algorithm that accounts for significant distance between agents in addition to distance from obstacles in their path, minimizing collision risk. Here we propose a multi-agent path planning algorithm which rewards paths with minimal cost-to-go and maximized distance between agents, minimizing the risk of error in path-following causing collision and thus damage to not one but two agents. This algorithm allows multi-agent systems to more effectively path plan and navigate around each other in an intuitive and cost-minimal way. We saw an increase of 2.015 units of distance on average between agents, which is significant in preventing collisions. This effect could be further improved by increasing the weight placed on distance-separation between agents.**

***Index Terms*— Multi-Agent Robotic Systems, Path Finding**

I. INTRODUCTION

Recent years have seen increasing research, both in industry and laboratory settings, on multi-agent systems, which have the potential to carry out much more complex tasks in larger spaces, such as warehouses and in video games. They are capable of covering larger areas, executing more nuanced tasks in less time, and allowing a diverse robotic skillset to complete a variety of tasks. They're also uniquely suited to dynamic environments with changing terrain, as each agent's set of tasks is smaller. However, with this increased functionality a new set of challenges are introduced, for path-finding, control, and task management alike. One of the largest path-finding challenges that is seen is collision avoidance - when several agents move dynamically on a shared field, the chance of potentially catastrophic collision rises exponentially. It is not merely enough to ensure the agents do not occupy exactly the same space - each agent requires a "buffer distance" around it to avoid collisions caused by imperfect control, error, and space constraints. This buffer distance is difficult to predict, however, as it depends on several factors including the speed of each agent, difficulty of the terrain, proximity to other static obstacles, extension of any appendages by either agent, and many more. Thus, it is most beneficial to ensure that each agent has a "buffer distance" around it that is as large as possible to prevent collisions. While collision avoidance has been maximized by several previous approaches (as described below), including avoiding high-density areas and imposing kinematic constraints, an algorithm has not yet been developed that prioritizes increased distance between agents and increases the penalty as the agents grow closer to one another. We present an algorithm that creates a maximum distance buffer around agents, potentially causing a longer path but ensuring that collision is avoided. This allows agents to more effectively navigate around each other and ensures that one agent's failure won't potentially lead to a domino effect of collisions.

A. Related Works: Collision Avoidance

Current research efforts into collision avoidance have primarily involved enforcing a minimum distance between agents and otherwise allowing them to move as they will. One key effort involved density mapping in crowded environments. This research focused on environments with several static and dynamic obstacles and rewarded taking a path through less densely populated areas, even if they were longer. This not only saved collisions, it also saved time, as agents could move quicker, stop less, and did not require perfect precision, showing that collision optimization can also lead to time optimization [1]. Another key effort was a study by Hoenig et. al. which imposed kinematic constraints on robots in close proximity to each other or static obstacles in addition to distance constraints (limited maximum linear and angular velocity). This led to more precise path following in these areas and allowed collision avoidance without decreasing velocity on clearer areas of the path [2]. These efforts and many similar others have taken strides towards eliminating collisions in multi-agent systems; however, they have not yet studied the effect of rewarding maximized distance between agents, as this study aims to do.

II. ENVIRONMENT SETUP AND PROBLEM STATEMENT

A. Environment

Consider a 2D 20x20 grid *M* populated with a team of *N* independent agents ranked by priority, as well as a collection of static obstacles. In this environment, the coordinate pair $(x_{A_{kt}}, y_{A_{kt}})$ denotes the coordinate location (x, y) of agent A_k at time *t* (given $0 \leq k \leq N$). Each of these cells has two possible states - empty or occupied. Occupied cells can be filled by one of two possible obstacles:

[1]Sahana Kumar is with the Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, U.S.A.
skumar78@jhu.edu

[2,3]Ishani Chatterjee, Tushar Kusnur, and Maxim Likhachev are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A. {ichatter, tkusnur,mlikhach}@andrew.cmu.edu

1) Dynamic obstacles or agents, which are specified by a start position and a goal position. Agents can move in four possible directions (up, down, left, right) or remain in place for each timestep. A step along any edge has a cost of one. Dynamic obstacles can occupy a cell for any set of times [i, j], where $0 \leq i \leq j \leq t$ and t is the total time required for all agents to reach their goal states. Each agent moves as determined by a modified A* prioritized planning algorithm (as described below).
2) Static obstacles, which remain in their position for the entirety of the time t. The positions occupied by static obstacles are unavailable to dynamic obstacles and steps that move onto them have a cost of infinity and thus cannot occur.

B. Prioritized Planning

Currently, the most common approach to multi-agent path planning is to use a prioritized planning system. This system first separates each agent (A_1-A_n) and ranks them based on priority. It then computes the path of the highest priority agent (for example, A_1) using our modified A* algorithm. It proceeds to compute the path of the next highest agent (A_2), using A_1's path as a dynamic obstacle that changes based on time (via a x-y-time plot). This would proceed as such until the final agent (A_n) is reached. The path of A_n would be computing using all agents A_1 to A_{n-1} as dynamic obstacles.

C. Cost Function

For the team of *N* agents described above in Section A: *Environment*, maximum distance between agents and thus minimum risk of collisions will be achieved by the maximizing the following function for each point in the path of each agent:

$$dist = \sum_{i=1}^{n}\sum_{k=1}^{n} \mid x_{A_{kt}} - x_{A_{it}} \mid + \mid y_{A_{kt}} - y_{A_{it}} \mid$$

III. METHODS

A. Algorithm

In order to effectively maximize inter-agent distance, we modify the g-value of each successor node s' to be smaller when the D(s'), sum of distances between s' and other agents at time t(s'), is greater. To prevent g(s') from being negative, we calculate some large constant c_moving(s, s') and subtract D(s') from it. This makes g(s') = g(s) + c_moving(s, s') - D(s'), as opposed to the g value in a normal A* algorithm, g(s') = g(s) + c(s, s'). This change is shown in the pseudo-code below, with the cost function differences highlighted in yellow. The algorithm depicted in Fig. 1 is used for the control paths shown in *Section IV: Results*, while the algorithm in Fig. 2 is used to derive the distance-maximized paths.

```
procedure PLANNER:
      c(s, s') ← constant edge cost
      g(s_start) = 0; g(s_other) = ∞
      insert s_start into OPEN
      while OPEN is not empty:
            remove s with smallest value f(s) from OPEN
            if position(s) == GOAL
                  break
            for each successor s' of s:
                  if s' not seen, g(s') = ∞
                  if g(s') > g(s) + c(s,s')
                        g(s') = g(s) + c(s,s')
                  if(s') ∉ CLOSED
                        insert s' into OPEN
      Generate path to GOAL by finding the origin of each node
```

Fig. 1. A* algorithm pseudocode

```
procedure DISTANCE_PLANNER:
      N ← number of higher-priority agents; A_k ← agent with priority k
      c_moving(s,s') = constant moving cost
      D(s') ← Σ_{i=1}^{N} dist(position(s'), position(A_i))
      g(s_start) = 0; g(s_other) = ∞
      insert s_start into OPEN
      while OPEN is not empty:
            remove s with smallest value f(s) from OPEN
            if position(s) == GOAL
                  break
            for each successor s' of s:
                  if s' not seen, g(s') = ∞
                  if g(s') > g(s) + c_moving(s, s') - D(s')
                        g(s') = g(s) + c_moving(s, s') - D(s')
                  if(s') ∉ CLOSED
                        insert s' into OPEN
      Generate path to GOAL by finding the origin of each node
```

Fig. 2. Modified A* algorithm with distance maximization

IV. RESULTS

A. Evaluation Criteria and Experimental Setup

In order to evaluate the distance-modified A* algorithm, we run a set of tests on a sample set of ten experiments (E1-E10). E1-E5 are populated by two dynamic agents, while E6-E10 are populated by four. E1, E2, E6, and E7 have no dynamic obstacles, E3, E4, E8, and E9 have large static obstacles composed of 5x5 square grids, and E5 and E10 both have cul-de-sacs. We run each experiment with a normal A* algorithm, as well as a distance-modified algorithm with weights of one to gauge the impact of the distance-modified algorithm. The only difference between environments E1 and E6, E2 and E7, E3 and E8, E4 and E9, and E5 and E10 are the number of agents - each pair has an identical obstacle setup. For each environment, we compare the total distance of the path and the average distance between each agent when using an A* algorithm to run it versus a distance-modified algorithm. This contrast allows us to evaluate the efficacy of this algorithm effectively. Each of these environmental setups are run in the environment *M* described in the above *Environment* section. The ten environments E1-10 are summarized in Fig 3 and 4 below, along with experimental results.

2-Agent Environments						
	Agent Paths		Metrics			
E1	*Agent 1 Start + Goal*	(0, 0) → (19,19)	*A* Average Dist.*	0.58	*Distance-Modified Average Dist.*	7.26
	Agent 2 Start + Goal	(1,0) → (19,18)	*A* Total Time*	43	*Distance-Modified Total Time*	43
E2	*Agent 1 Start + Goal*	(0,0) → (15,17)	*A* Average Dist.*	8.24	*Distance-Modified Average Dist.*	8.35
	Agent 2 Start + Goal	(9,5) → (1, 18)	*A* Total Time*	34	*Distance-Modified Total Time*	34
E3	*Agent 1 Start + Goal*	(0,0) → (15,17)	*A* Average Dist.*	8.88	*Distance-Modified Average Dist.*	11.92
	Agent 2 Start + Goal	(18,14) → (4,3)	*A* Total Time*	34	*Distance-Modified Total Time*	51
E4	*Agent 1 Start + Goal*	(17,17) → (2, 4)	*A* Average Dist.*	6.9	*Distance-Modified Average Dist.*	6.72
	Agent 2 Start + Goal	(9,13) → (15,1)	*A* Total Time*	30	*Distance-Modified Total Time*	30
E5	*Agent 1 Start + Goal*	(4,3) → (18,10)	*A* Average Dist.*	3.92	*Distance-Modified Average Dist.*	3.8
	Agent 2 Start + Goal	(5,8) → (16,19)	*A* Total Time*	25	*Distance-Modified Total Time*	25

Fig. 3. Experiments E1-E5 (two-agent experiments) with their agent paths and experimental data.

4-Agent Systems								
	Agent Paths				Metrics			
E6	*Agent 1 Start + Goal*	(1,2) → (0,18)	*Agent 3 Start + Goal*	(2,6) → (13,16)	*A* Average Dist.*	10.07	*Distance-Modified Average Dist.*	12.08
	Agent 2 Start + Goal	(3,16) → (17,17)	*Agent 4 Start + Goal*	(19,19) → (13,12)	*A* Total Time*	23	*Distance-Modified Total Time*	23
E7	*Agent 1 Start + Goal*	(11,19) → (0,0)	*Agent 3 Start + Goal*	(19,2) → (11,2)	*A* Average Dist.*	7.94	*Distance-Modified Average Dist.*	8.49
	Agent 2 Start + Goal	(4,18) → (14,8)	*Agent 4 Start + Goal*	(8,19) → (7,12)	*A* Total Time*	32	*Distance-Modified Total Time*	32
E8	*Agent 1 Start + Goal*	(15,2) → (1,18)	*Agent 3 Start + Goal*	(15,17) → (3,6)	*A* Average Dist.*	4.51	*Distance-Modified Average Dist.*	5.58
	Agent 2 Start + Goal	(11,19) → (4,2)	*Agent 4 Start + Goal*	(13,5) → (9,19)	*A* Total Time*	35	*Distance-Modified Total Time*	48
E9	*Agent 1 Start + Goal*	(17,3) → (18,18)	*Agent 3 Start + Goal*	(5,7) → (3,18)	*A* Average Dist.*	7.65	*Distance-Modified Average Dist.*	11.13
	Agent 2 Start + Goal	(11,19) → (16,19)	*Agent 4 Start + Goal*	(14,7) → (5,19)	*A* Total Time*	23	*Distance-Modified Total Time*	55
E10	*Agent 1 Start + Goal*	(4,3) → (18,18)	*Agent 3 Start + Goal*	(18,5) → (3,18)	*A* Average Dist.*	6.78	*Distance-Modified Average Dist.*	10.29
	Agent 2 Start + Goal	(10,10) → (16,19)	*Agent 4 Start + Goal*	(5,6) → (5,19)	*A* Total Time*	33	*Distance-Modified Total Time*	57

Fig. 4. Experiments E6-E10 (four-agent experiments) with their agent paths and experimental data.

B. Experimental Results

In running this sample set of environments, it was seen that on average, the addition of the distance cost function significantly increased the distance between agents. On average, the distance between agents was increased by 2.015 units, which is significant in terms of ensuring lack of collision. This effect was, as expected, lessened in environments where the inter-agent distance was greater than 5 units even when using solely an A* algorithm. This effect was particularly pronounced in examples where an A* algorithm found nearly identical paths, such as E1 (Fig. 5,6).

In contrast, two of the experiments with only 2 agents (E4 and E5) and a high number of obstacles experienced slightly lower inter-agent distance (less that 0.2 units of difference) with the distance-modified algorithm than with the normal A* algorithm. An example of this is shown in Fig. 7 and 8. This is because by minimizing distance at one step, the agent could potentially force itself into a more confined space in the next step. While this might matter less in a more open environment, when a high proportion of grid spaces are occupied, this leads to a suboptimal algorithm. Further optimization of the cost function is necessary to mitigate this issue. Experimenting with increasing the weight on the distance function did improve the problem, but resulted in its own issues, increasing the time taken to complete the path almost threefold.

V. CONCLUSIONS

The integration of distance-maximization into the A* cost functions allows agents to find safer paths with less risk of collision. Our sample set of experiments have demonstrated a significant increase in inter-agent distance, significantly minimizing collision possibility. Future steps include running the algorithm on a greater set of sample environments that are significantly larger, refining the algorithm to include more advanced search methods, such as Conflict-Based Search, and including distance from static obstacles in the cost function.

ACKNOWLEDGMENT

This work was made possible by the support and mentorship of the Likhachev lab, as well as the extensive network of resources and mentors provided by the Robotics Institute Summer Scholars (RISS) program. This material is based

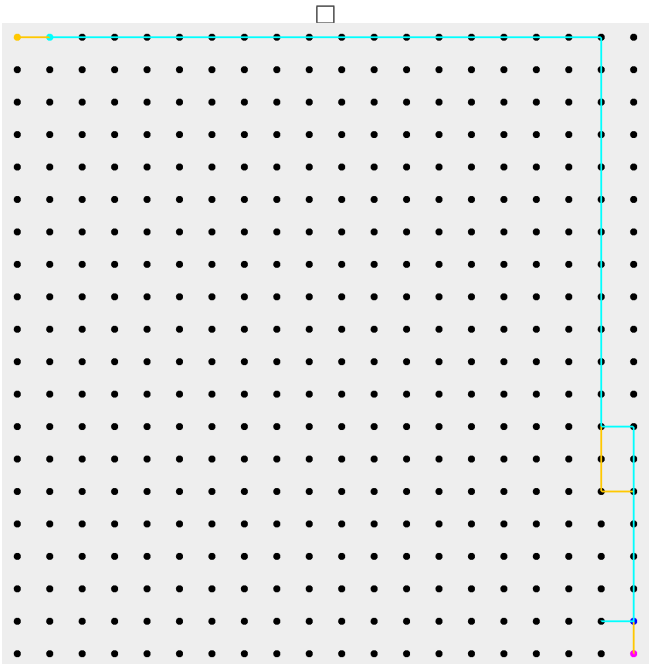


Fig. 5. Experiment E1 run with an A* planner. The orange lines represent the higher priority agent A1, with the orange dot at position (0,0) representing the start location and the magenta dot at position (19,19) representing the goal location. The cyan lines represent the lower priority agent A2, with the cyan dot at position (1,0) representing the starting position and the blue dot at position (19,18) representing the ending position.

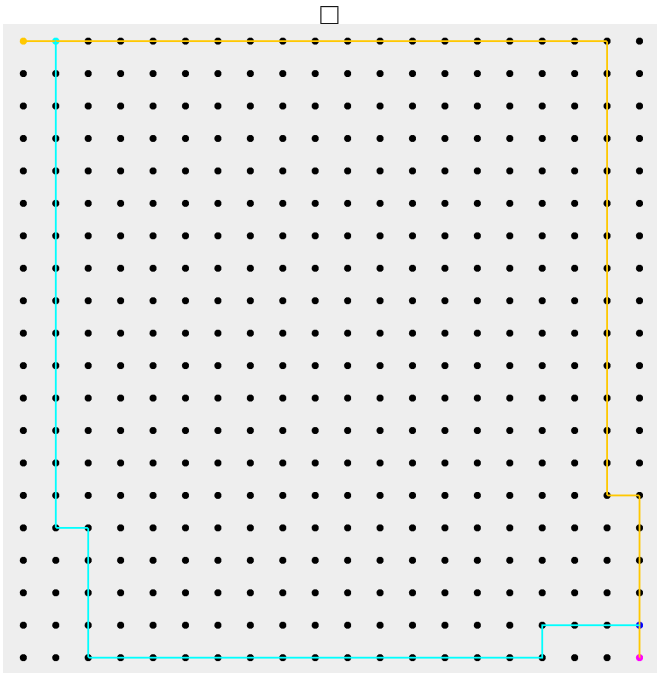


Fig. 6. Experiment E1 run with our distance-modified planner. The orange lines represent the higher priority agent A1, with the orange dot at position (0,0) representing the start location and the magenta dot at position (19,19) representing the goal location. The cyan lines represent the lower priority agent A2, with the cyan dot at position (1,0) representing the starting position and the blue dot at position (19,18) representing the ending position.

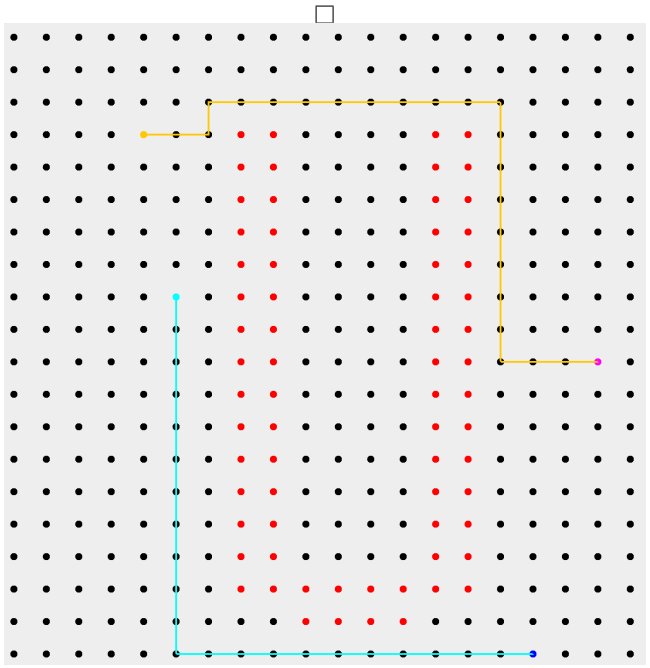


Fig. 7. Experiment E5 run with an A* planner. The orange lines represent the higher priority agent A1, with the orange dot at position (4,3) representing the start location and the magenta dot at position (18,10) representing the goal location. The cyan lines represent the lower priority agent A2, with the cyan dot at position (5,8) representing the starting position and the blue dot at position (16,19) representing the ending position. Each red dot represents a static obstacle.

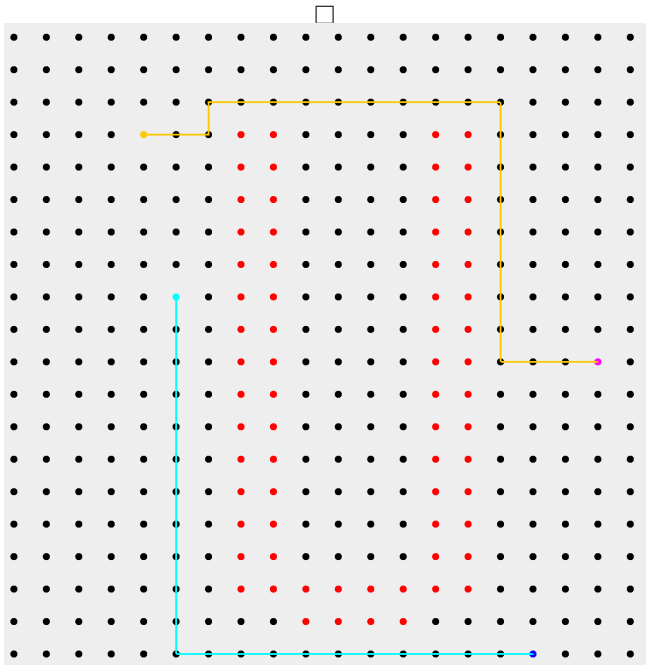


Fig. 8. Experiment E5 run with our distance-modified planner. The orange lines represent the higher priority agent A1, with the orange dot at position (0,0) representing the start location and the magenta dot at position (19,19) representing the goal location. The cyan lines represent the lower priority agent A2, with the cyan dot at position (1,0) representing the starting position and the blue dot at position (19,18) representing the ending position. Each red dot represents a static obstacle.

upon work supported by the National Science Foundation

under Grant No. 1659774.

REFERENCES

[1] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," in *Artificial Intelligence*, S. T. P. Doherty, Ed., 2015, vol. 219, pp. 40–66.

[2] D. Sigurdson, V. Bulitko, W. Yeoh, C. Hernández, and S. Koenig, *Multi-Agent Pathfinding with Real-Time Heuristic Search*. 2018 IEEE Conference on Computational Intelligence and Games (CIG), 2018, pp. 1–8.

[3] S. Bhattacharya, M. Likhachev, and V. Kumar, *Multi-agent path planning with multiple tasks and distance constraints*. Anchorage, AK: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 953–959.

[4] M. Hudziak, I. Pozniak-Koszalka, L. Koszalka, and A. Kasprzak, "Comparison of algorithms for multi-agent pathfinding in crowded environment," 2015.

[5] W. Hoenig, T. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints," 2017.

[6] E. Boyarski, A. Felner, D. Harabor, L. Stuckey, Peter Cohen, J. Li, and S. Koenig, "Iterative-deepening conflict-based search." 2020.

[7] D. Silver, "Cooperative pathfinding," Rocky Mountain Research Lab., Boulder, CO, pp. 117–122, May 2005.

[8] K.-H. Wang and A. Botea, "Fast and memory-efficient multi-agent pathfinding."

[9] T. Kusnur and S. Mukherjee, *A Planning Framework for Persistent, Multi-UAV Coverage with Global Deconfliction*, 2019.

[10] H. Ma, "Searching with consistent prioritization for multi-agent path finding," Dec. 2018.

Legibility of Path Trajectories Personalized for Multiple Perspectives

Ellen Mamantov[1], Ada Taylor[2], and Henny Admoni[2]

***Abstract*— Intent-expressive (legible) motion is a key component of human-robot interaction because it allows observers to make inferences about a robot's goals. Most algorithms that generate legible motion assume that observers have an omniscient perspective, where viewers see every change in the robot's configuration as it moves. However, there are potential benefits to personalizing the robot's motion for an observer's specific perspective. Furthermore, there are frequently multiple people sharing a space with a robot that all would benefit from being able to infer the intent of that robot. In this work, we present a model that plans navigation path trajectories with multiple observers' perspectives in mind. We test our model through video-based user studies that compare the legibility of paths personalized for omniscient, individual, and multiple perspectives. Our pilot user study did not provide convincing evidence that incorporating multiple perspectives into path planning increases legibility, but the results will inform the design of a full user study that will be conducted in the future.**

***Index Terms*— Social Human-Robot Interaction, Motion and Path Planning**

I. INTRODUCTION

People are well versed in understanding the intent of another human's actions, but it is much more difficult for a human to understand the goals of a robot. Therefore, the robot is responsible for conveying its intent to the humans around it. For example, in a restaurant, people are intuitively aware of other patrons and servers. Patrons can alter their behavior to best coordinate with servers, such as timing a break in their conversation for when their server arrives at their table. Underlying this ability is the fact that the patrons can infer the goals and intentions of the server.

A robot needs to be able to convey its intentions to its observers, and one way to do so is to make its motion legible (intent-expressive). However, current state-of-the-art path planning algorithms do not take into account the perspectives of the robot's observers, particularly when there are multiple viewers with different perspectives. Planning movement with observers' perspectives in mind has the potential to make a robot's motion more legible overall. In turn, because people will be better able to understand the robot's intentions and goals, their interactions with the robot should be safer and more rewarding.

In this paper, we study a robot's navigation path trajectories in a complicated scenario with multiple observers: a restaurant. The goal is to generate approach trajectories for a robot server that are more human-understandable and informative to the customers in the restaurant. We present a method for generating navigation paths for a robot server that takes into account the relative visibility of the path. We test the effect of taking into account multiple perspectives when planning path trajectories through an online, video-based human subjects experiment. Our study compares the legibility of trajectories that were personalized for one perspective or two perspectives to baseline trajectories that were planned under the assumption that observers have an omniscient perspective. We hypothesize that:

H1 A robot that plans its path taking into account the visibility of that path to all of its observers can create paths in which the goal is easier for viewers to understand.

H2 A path personalized for a specific perspective will be more legible than the average multi-perspective performance for that perspective but will be less legible for other perspectives.

Our preliminary pilot study failed to provide support for either hypothesis and the results suggested that the paths were not distinct enough to be more or less legible than each other. However, the pilot study will effectively inform the design of the full user study that will be conducted in the future.

II. BACKGROUND

Plenty of previous studies have focused on legible motion that conveys the robot's intention to the observer(s). It has been shown that people are better at inferring an approaching agent's end destination when the agent is a human rather than a robot [1]. One solution that has been studied is adding behaviors to a robot's motion that aid in people's understanding of the robot. It has been shown that adding anticipatory gestures to a robot's motions increased people's ability to identify the robot's intent [2]. Similarly, adding forethought and reaction animations increased the readability of robots and improved people's perceptions of them [3]. However, these solutions are all explicit ways to express intent. In many situations, it is valuable for a robot to be able to convey its intent implicitly [4].

An implicit solution that has received attention in the literature is changing robots' motion trajectories to help people understand what the robot is trying to accomplish, but the findings have been mixed. While one study found that people prefer a robot to approach them from the side rather than approach straight at them from the front [5], another study concluded that straight trajectories are best [6]. Some work has found that predictability and legibility are correlated characteristics of motion and have treated

[1]Ellen Mamantov is with the Computer Science and Psychology Departments, Carleton College, Northfield, MN, USA mamantove@carleton.edu

[2]Ada Taylor and Henny Admoni are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA {hadmoni, adat}@andrew.cmu.edu

legibility and predictability as inseparable [6]–[9]. Other work has found that legibility and predictability are actually separate, sometimes contradictory, characteristics [10]–[12]. It has been shown that path trajectories that do not match humans' expectations but convey the motion's goal or intent to a human observer more successfully exist. These less-predictable and more-legible paths allow a human observer to infer the robot's intent confidently and quickly [10].

Past research involving legibility of motion has utilized a wide variety of different methods. In some, people interacted with robots in person [1], [5], [12], [13], while in others people viewed videos of a robot moving or a simulation of a robot's motion [2], [8]–[11], [13]. The robot's tasks also vary widely among different studies and include picking up one of two cups [10], moving to one of three tables [1], delivering a TV remote using three different paths [13], and using different trajectories to avoid a collision while crossing paths with a human [9]. However, despite this wide variety of studies, very few have considered an observer's perspective when studying legibility, even though past work has identified an observer's viewpoint as one important aspect that affects whether a robot's motion is legible [11]. Most contemporary legible motion planners assume that human observers are omniscient and are aware of all changes in the robot's configuration as it moves, even though human observers have a particular viewpoint that does not encompass the entire scene. One previous study investigated this assumption and introduced a model that optimizes motions with a certain 2D-projection in mind; it was shown that certain viewing angles can lead to depth uncertainty and occlusions that make certain trajectories less legible [14]. However, this technique does not deal with the overall field of view or multiple perspectives, but instead focuses on how motions within a field of view are perceived.

In this paper, we consider legible motion in a more complicated scenario with multiple observers. While past work has considered the effect of viewpoint on the legibility of a robot's arm movements [11], [14], we focus on the legibility of navigation paths that a robot takes to approach targets. For example, in a restaurant, multiple people that are sitting at one table should all be able to recognize a server's intended destination. Personalizing path trajectory for one of the diners may lead to the other diners being unable to infer the server's goal. To address this, we present a method for planning paths that are personalized for multiple observers' perspectives. We test the effect of taking into account multiple perspectives when planning approach trajectories through a video-based human subjects experiment. Our study compares the legibility of trajectories that were personalized for one perspective or two perspectives to a baseline of an omniscient perspective.

III. METHODS

To test our method for creating personalized paths for observers, we conducted a pilot user study and plan to conduct a full-scale user study after making adjustments based on the results and feedback gathered during the pilot. Therefore, this work remains in progress. According to our first hypothesis, we expect that taking into account multiple perspectives will create more legible paths than taking into account a single perspective and that taking into account a single perspective will generate more legible paths than assuming an omniscient perspective. According to our second hypothesis, we also expect that a path personalized for a specific observer's perspective will be more legible for that person than the paths generated with all perspectives or no perspectives in mind. However, we also expect that a specific observer will find a path generated for a different observer less legible than a path generated with both perspectives in mind, and the multiple-perspective path will out-perform the incorrectly personalized one.

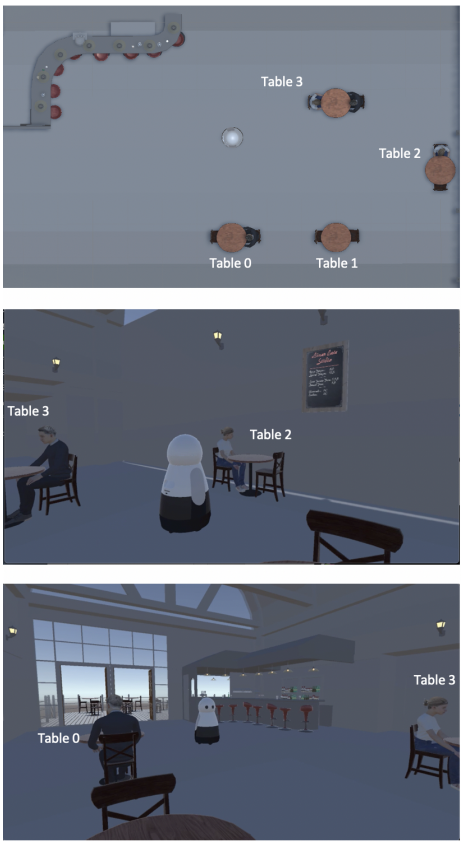


Fig. 1. The restaurant scene from above (top) and from the two perspectives that participants could be assigned to: Perspective A (middle) and Perspective B (bottom). Perspective A and B are "sitting" at Table 1

A. Task

Participants were shown videos of a robot server approaching different "goal tables" in the restaurant. They were tasked with indicating how confident they are that the server is approaching their table.

B. Stimuli

We planned our video stimuli by manipulating several variables. In every video, a robot moves from a starting location to one of four goal tables: the participant's table (table 1) or a table in front of (table 0), across from (table

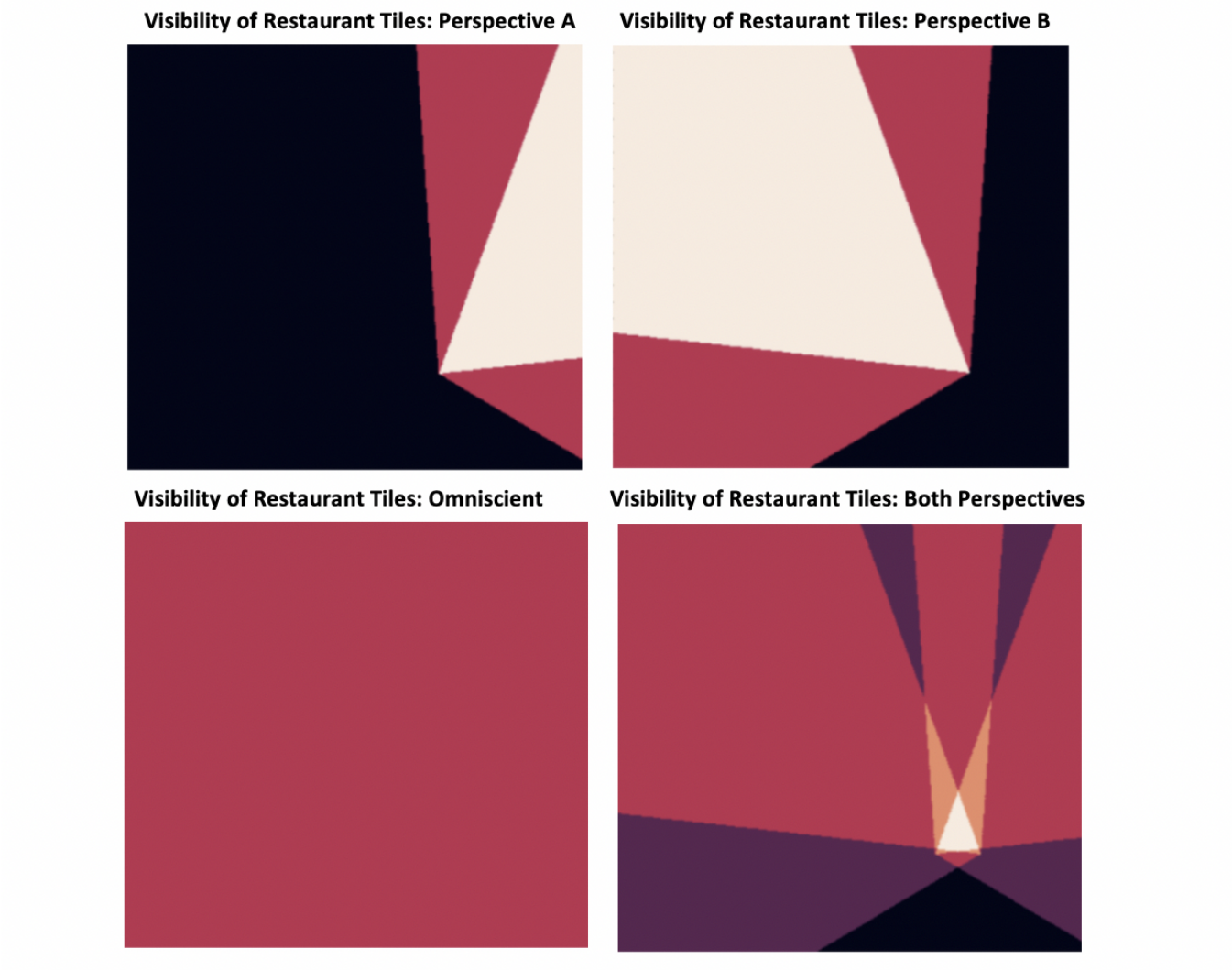


Fig. 2. The cost functions that correspond to the four path planning method conditions. Each one is a map of the visibility of one of the four sets of perspectives

3), or perpendicular to (table 2) the participant's table. Given this basic setup, we manipulate:

Perspective, by changing which of the two chairs at the participant's table they are watching the scene from;

Path Planning Method, by having the robot follow a path planned for an omniscient perspective, both perspectives, an individual perspective (A), or the other individual perspective (B).

Given that there are four goal tables, four path planning methods, and two perspectives, we had a total of 32 video stimuli.

C. Path Planning

Each of the four path planning methods was used to generate paths for each of the four goal locations, creating 16 in all. All of the paths begin at the same starting position and consisted of seventeen waypoints. All were created with the same table layouts and observers, though not all of the cost functions for assessing paths take these into account.

For each of the four path planning methods, we generated a map of the visibility of each set of observers accounted for in that path planning method. The four maps are shown in Figure 2. For the omniscient condition, we did not record any visibility. For each single-observer condition, we projected a 60° cone in the direction of the observer's perspective and another 120° cone representing their peripheral vision. Points within the first cone were given a reward of 1 unit of vision, points within the outer cone were given 0.5 units of vision. For the Both Perspectives condition, we combined the two individual mappings. A mapping of the cost to traverse points in the scene was also generated, with obstacle points such as tables given an infinite cost, and a buffer zone around each to account for the navigation of the robot that was treated as part of the obstacle.

To generate paths, we created bezier curves anchored at the beginning and ending points of each path, sampling additional control points from within the cones of vision. From these generated paths, we chose those with the lowest

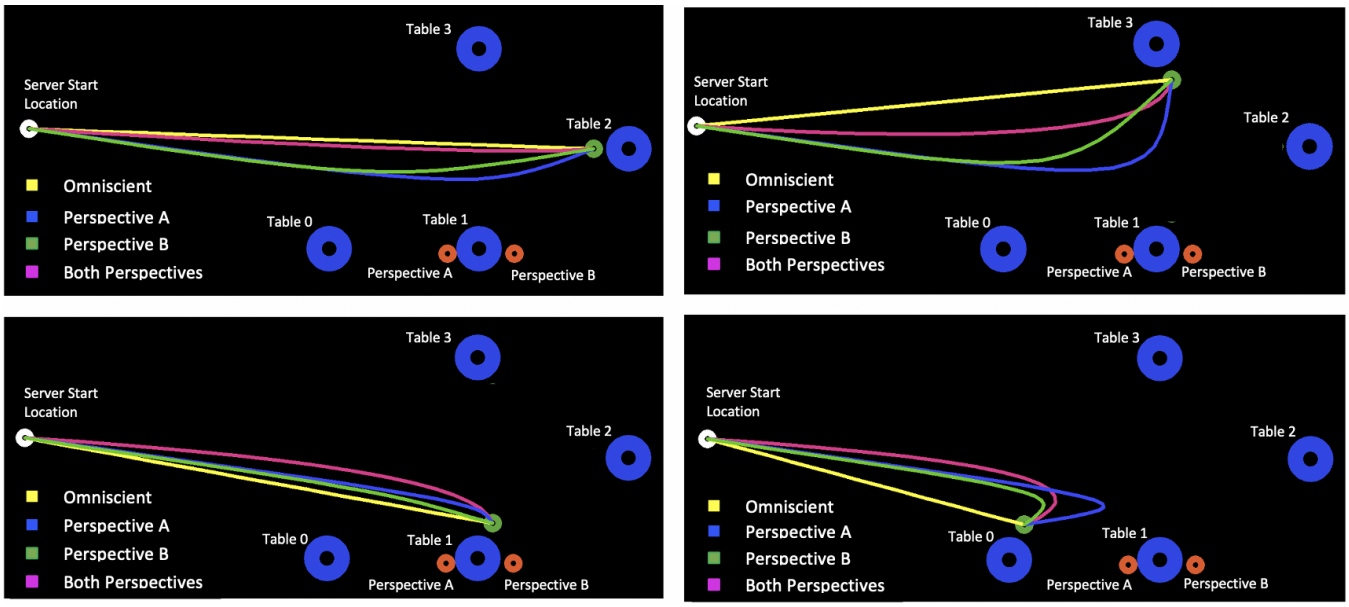


Fig. 3. The paths to each of the four tables in the restaurant that were used in the user study. Blue paths were personalized for Perspective A. Green paths were personalized for Perspective B. Pink paths were optimized while considering both observers' perspectives, and yellow paths were not personalized for either perspectives.

cost, as defined by:
For a path ξ with weight on the visibility function k (set to 3 for our experiment):

$$C_\xi = \int_\xi C_{\text{obstacles}}(p) + C_{\text{distance}}(p) + k * C_{\text{visibility}}(p)$$

Where

$$C_{\text{obstacles}}(p) = \begin{cases} \infty, & \text{if obstacle map shows } p \text{ blocked} \\ & \text{by obstacle or buffer} \\ 0, & \text{otherwise} \end{cases}$$

$$C_{\text{distance}}(p) = 1$$

Given our map of the visibility from each point as V:

$$C_{\text{visibility}}(p) = \frac{\max(V) - V(p)}{\max(V)}$$

Thus, the total cost of a given path is the integral along the path, or the sum of these values at every point. Among the generated paths, we chose those with the lowest overall cost for the pilot.

Intuitively, for the omniscient point of view, this results in no waypoints and no benefit to warping the path from a straight line, and therefore a straight line. Paths which dip into the vision areas are longer, and their lower cost in $C_{\text{visibility}}$ to compensates for the higher cost in C_{distance}.

D. Procedure

We developed a restaurant simulator using the Unity Game Engine software. The robot server was a model of the Kuri Robot. We generated the stimuli by recording the simulator as it played. The stimuli only differed by the path waypoints that were supplied for the robot to follow and which of the two perspectives it was recorded from.

The study was constructed and deployed online using the open-source framework psiTurk [15] and consisted of a tutorial that introduced the participants to the restaurant's layout and the mechanics of the slider that they would use to report their confidence, the trials, and a final questionnaire. To ensure that no participant saw the same path twice (but from different perspectives), each participant was randomly assigned to one of the two perspectives and shown 16 video stimuli (4 goal tables x 4 pathing methods). We randomized the order of the stimuli to avoid ordering biases.

During the trials, participants used a continuous slider to report how confident they were that the server was approaching their table on a scale from 0 to 100. They reported their confidence continuously, and the video would only continue playing if they were actively holding the slider. The participants were not able to report their confidence after the videos ended, which occurred when the server reached its goal.

The post-study questionnaire included open-ended questions that allowed participants to provide feedback about the task.

E. Participants

We recruited 20 participants from the community (6 male, 12 female, 2 unspecified, aged 21-29) for a pilot study. Ten participants were randomly assigned to the Perspective A condition, and the other ten participants were assigned to the Perspective B condition.

IV. Analysis and Results

A. Raw Confidence Data

Confidence data were collected continuously during each trial. Participants reported how confident they were that the server was approaching their table on a scale of 0 to 100. Reporting a low confidence value also meant that they were very confident that the server was approaching a different table. Figure 4 provides an example of this data.

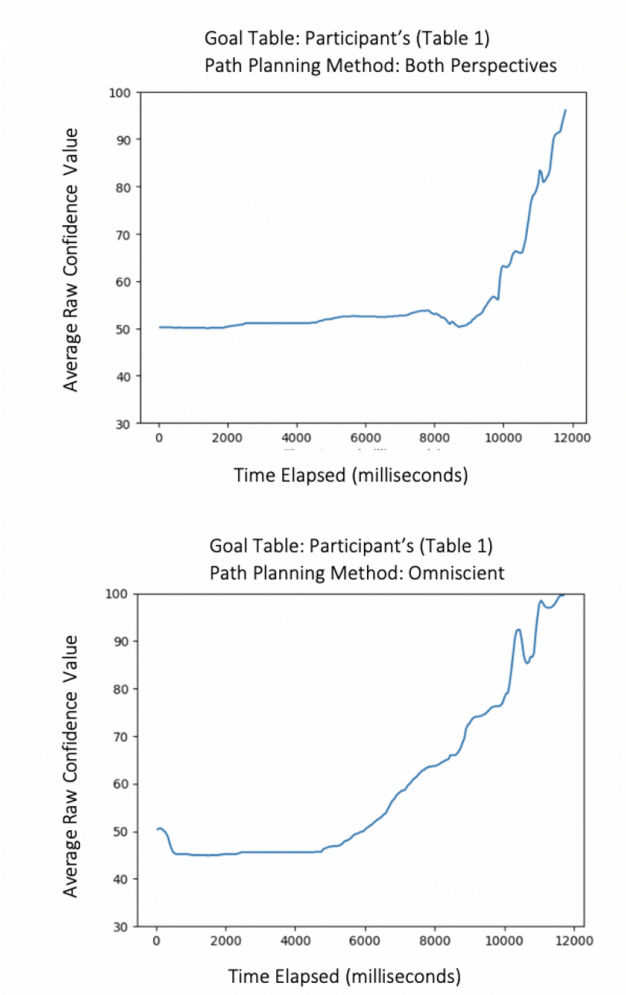


Fig. 4. Average raw confidence score over time during two of the sixteen trials shown to participants that were watching from Perspective B: the trial during which the robot approached the participant's table (table 1) using the omniscient path planning method (Bottom) and the trial during which the robot approached the participant's table (table 1) using the both perspectives path planning method (Top).

B. Testing Hypothesis 1

To test our first hypothesis, we compared the performance of participants across the four different path planning methods conditions. Because hypothesis 1 focuses on the overall performance of all participants, regardless of which perspective they are viewing from, we did not keep participants separated by their perspective condition for this analysis. We analyzed three different aspects of performance: continuous correctness, accuracy, and number of reversals.

1) Continuous Correctness: For analysis, we altered the raw confidence data such that a value of 100 represents high confidence that the server is approaching the *correct* table, whether or not the correct table was the participant's table. We accomplished this by simply flipping the scale of the raw confidence data whenever the goal table was not the participant's table. We refer to these values as continuous correctness values. An average continuous correctness value was calculated for every trial, with continuous correctness values weighted by the amount of time the participant reported that value. A one-way analysis of variance (ANOVA) was calculated on the participants' average continuous correctness value. The analysis was not significant, ($f(3) = 0.555$, $p = 0.645$).

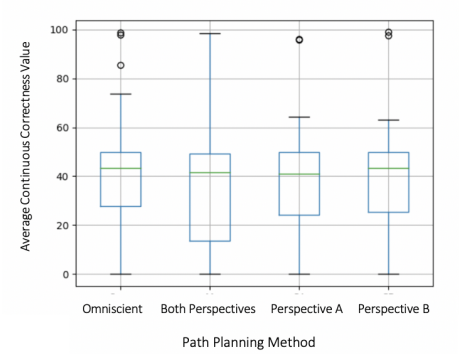


Fig. 5. Box plot showing the distribution of participants' average continuous correctness values by path planning method condition

2) Accuracy: Accuracy values are discrete values based on the raw confidence data. The participant was considered accurate if their raw confidence value was over 55 and the server's goal was their table or if their confidence was under 45 and the server's goal was a different table. They were considered unsure if their confidence was between 45 and 55. An average accuracy score was calculated for every trial. An accurate confidence value received a score of 1, and inaccurate confidence value received a score of 0, and an unsure confidence value received a score of 0.5. To calculate the average accuracy of the trial, each score was weighted by the amount of time the participant was correct, incorrect, or unsure, respectively. A one-way analysis of variance (ANOVA) was calculated on the participants' average accuracy value. The analysis was not significant, ($f(3) = 0.454$, $p = 0.715$).

3) Reversals: We defined a reversal as a participant changing their prediction about the server's goal. Reversal data was also calculated from the raw confidence data. A reversal occurred each time a participant's confidence score changed from below 45 to above 55 or from above 55 to below 45. The total number of reversals was counted for each trial, and the average number of reversals for each path planning method was calculated. A one-way analysis of

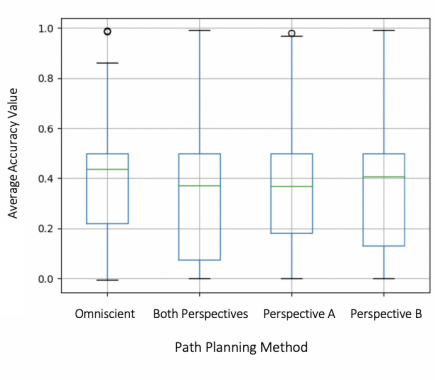


Fig. 6. Box plot showing the distribution of participants' average accuracy values by path planning method condition

variance (ANOVA) was calculated on the participant's average number of reversals. The analysis was not significant, ($f(3) = 0.503$, $p = 0.681$).

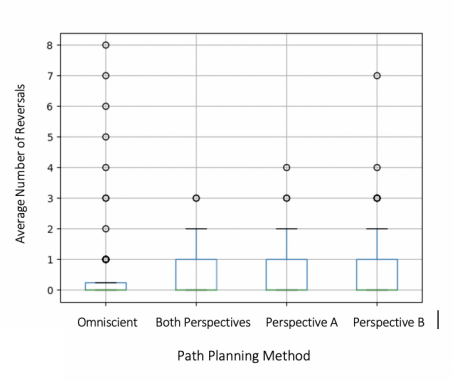


Fig. 7. Box plot showing the distribution of participants' average number of reversals by path planning method condition

Our first hypothesis was not supported by the pilot study results due to the lack of significant improvement in participant performance in trials that presented paths that were personalized for both perspectives.

C. Testing Hypothesis 2

To test our second hypothesis, we compared the performance of participants during trials that presented paths that were personalized for the perspective they were watching from ("matched trials") to their performance during trials that presented paths that were personalized for the perspective they were not watching from ("mismatched trials"). In other words, matched trials were those in which participants in Perspective A watched paths personalized for Perspective A or participants in Perspective B watched paths personalized for Perspective B. Mismatched trials were those in which participants in Perspective A watched paths personalized for Perspective B and participants in Perspective A watched paths personalized for Perspective B. We used the same methods that are described above to analyze continuous correctness, accuracy, and number of reversals.

1) Continuous Correctness: A one-way analysis of variance (ANOVA) was calculated on participants' average continuous correctness value during either matched or mismatched trials. The analysis was not significant, ($f(1) = 0.137$, $p = 0.711$).

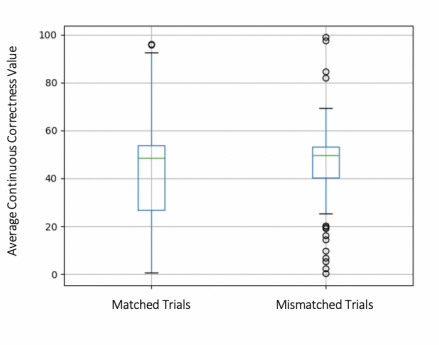


Fig. 8. Box plot showing the distribution of participants' average continuous correctness values in matched trials (perspective and personalization match) or mismatched trials (perspective and personalization differ)

2) Accuracy: A one-way analysis of variance (ANOVA) was calculated on participants' average accuracy value during either matched or mismatched trials. The analysis was not significant, ($f(1) = 1.194$, $p = 0.276$).

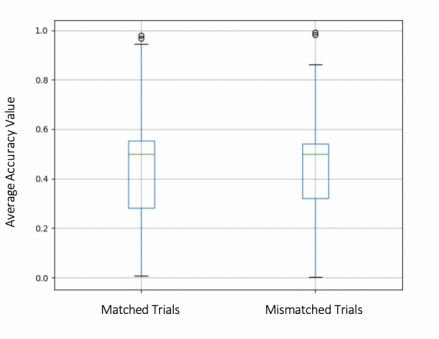


Fig. 9. Box plot showing the distribution of participants' average accuracy values in matched (perspective and personalization match) or mismatched trials (perspective and personalization differ)

3) Reversals: A one-way analysis of variance (ANOVA) was calculated on participants' average number of reversals during either matched or mismatched trials. The analysis was not significant, ($f(1) = 0.295$, $p = 0.588$).

Our second hypothesis was also not supported by the pilot study results because there are no significant improvements in participant performance in matched trials.

D. Implications for Future Study

Despite the lack of significant findings, the pilot user study described here will be used to inform the development of a full-scale user study that will be conducted in the future. Qualitative feedback from pilot study participants indicates

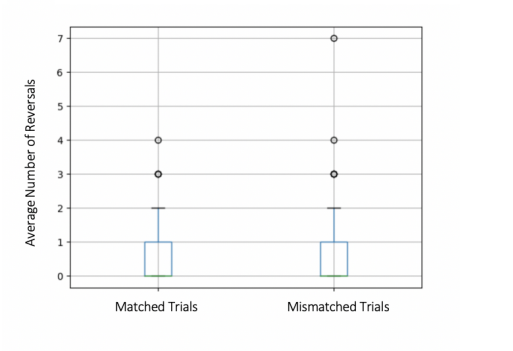


Fig. 10. Box plot showing the distribution of participants' average number of reversals in matched (perspective and personalization match) or mismatched trials (perspective and personalization differ)

that they were not able to become comfortable with the input range slider that we utilized to collect raw confidence data during the task. They noted that the tutorial felt short and wished it was more related to the real study's task. Moving forward, we plan to incorporate a longer tutorial that not only introduces participants to the mechanics of the confidence slider but also to the specific task of inferring where a server is approaching. Their feedback also suggests that more learning occurs throughout the task than we originally planned for. Participants reported that they adjusted to the task as it went on. To mitigate practice effects, we plan to use a Latin squares design in the future study rather than simply randomizing the presentation of the stimuli.

Other notable qualitative feedback includes the fact that the paths appeared extremely similar to participants during the study. Multiple participants noted that they thought that stimuli were being repeated, even though each stimulus was unique. The lack of statistically significant results, in combination with the participants' feedback, suggests that the paths were not distinct enough from each other to affect participants' confidence in their inferences. We plan to change portions of our path generation process to adjust for this. To help us ensure that the paths we generate are noticeably distinct, we plan to develop metrics that will allow us to quantitatively measure how different two paths, or two sets of paths, are.

Finally, we have realized that our performance metrics and data analysis may not fully capture differences that exist between the path planning methods. We found that performance depended on which condition the participant was in because viewing the scene from one perspective (Perspective A) was much more difficult than viewing it from the other perspective (Perspective B). However, our data analysis relied on averaging over the lengths of trials and across all participants, regardless of perspective condition. Thus, we may have failed to identify smaller, but still significant, trends. We plan on incorporating more nuanced analysis into our future work.

V. CONCLUSION

In this paper, we have proposed a change to path planning with the goal of improving the legibility of navigation paths taken by robots to approach people. We show that taking into account multiple observers' perspectives when creating trajectories results in paths that differ from ones that are created when only taking into account one perspective or ignoring observers' perspectives entirely. We developed a user study to test the legibility of different paths using a simulated restaurant scenario and piloted it to discover its strengths and weaknesses. The lack of significant results in the pilot user study simply informs changes that will be made before the full user study is conducted.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774 and the Sony Corporation. We would like to thank the organizers of the Carnegie Mellon Robotics Institute Summer Scholars (RISS) program for supporting this experience and everyone at the Human and Robot Partners Lab at Carnegie Mellon University for their guidance and support.

REFERENCES

[1] P. Basili, M. Huber, O. Kourakos, T. Lorenz, T. Brandt, S. Hirche, and S. Glasauer, "Inferring the goal of an approaching agent: A human-robot study," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 527–532.

[2] M. J. Gielniak and A. L. Thomaz, "Generating anticipation in robot motion," in *2011 RO-MAN*. IEEE, 2011, pp. 449–454.

[3] L. Takayama, D. Dooley, and W. Ju, "Expressing thought: improving robot readability with animation principles," in *Proceedings of the 6th international conference on Human-robot interaction*, 2011, pp. 69–76.

[4] E. Cha, Y. Kim, T. Fong, M. J. Mataric, *et al.*, "A survey of nonverbal signaling methods for non-humanoid robots," *Foundations and Trends® in Robotics*, vol. 6, no. 4, pp. 211–323, 2018.

[5] D. Bortot, M. Born, and K. Bengler, "Directly or on detours? how should industrial robots approximate humans?" in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 89–90.

[6] M. Beetz, F. Stulp, P. Esden-Tempski, A. Fedrizzi, U. Klank, I. Kresse, A. Maldonado, and F. Ruiz, "Generality and legibility in mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, p. 21, 2010.

[7] C. Lichtenthäler and A. Kirsch, "Legibility of robot behavior: a literature review," 2016.

[8] ——, "Towards legible robot navigation-how to increase the intend expressiveness of robot navigation behavior," 2013.

[9] ——, "Goal-predictability vs. trajectory-predictability: which legibility factor counts," in *Proceedings of the 2014 acm/ieee international conference on human-robot interaction*, 2014, pp. 228–229.

[10] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 301–308.

[11] R. M. Holladay, A. D. Dragan, and S. S. Srinivasa, "Legible robot pointing," in *The 23rd IEEE International Symposium on robot and human interactive communication*. IEEE, 2014, pp. 217–223.

[12] A. D. Dragan, S. Bauman, J. Forlizzi, and S. S. Srinivasa, "Effects of robot motion on human-robot collaboration," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2015, pp. 51–58.

[13] K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, C. L. Nehaniv, A. Sisbot, R. Alami, and T. Siméon, "How may i serve you? a robot companion approaching a seated person in a helping context," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 172–179.

[14] S. Nikolaidis, A. Dragan, and S. Srinivasa, "based legibility optimization," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 271–278.
[15] T. M. Gureckis, J. Martin, J. McDonnell, A. S. Rich, D. Markant, A. Coenen, D. Halpern, J. B. Hamrick, and P. Chan, "psiturk: An open-source framework for conducting replicable behavioral experiments online," *Behavior research methods*, vol. 48, no. 3, pp. 829–842, 2016.
[16] A. Dragan and S. Srinivasa, "Familiarization to robot motion," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, 2014, pp. 366–373.
[17] C. Lichtenthäler, T. Lorenz, and A. Kirsch, "Towards a legibility metric: How to measure the perceived value of a robot," 2011.

Destination-Aware Auction Systems for Paratransit Scheduling

Rebecca Martin[1], Lan Wu[2], Yu Wu[3], Isaac Isukapati[4], and Stephen Smith[4]

***Abstract*— Paratransit systems provide an equitable transit service to the elderly and handicapped through shared-ride, door-to-door transportation. The main objective of the dial-a-ride problem (DARP) is to optimally schedule and dispatch available vehicles to satisfy ride requests between given pick-up and drop-off locations at specified times. One type of solution to this scheduling problem utilizes an auction system to assigning requests to tasks. However, in these static systems, all ride requests are known upfront, and cannot accommodate any last minute or real time requests. In addition, these systems do not take into account that the time costs associated with each request are destination dependent. To remove this rigidity and equal prioritization of requests, our research schedules requests dynamically and assigns priorities to requests based on their destination. To evaluate our algorithms, we simulated our system in Python and compared it against a static optimization algorithm [1]. Overall, the auction-based system performed well, scheduling at least 95% of the requests more efficiently.**

***Index Terms*— Paratransit, Dial-A-Ride Problem, Multi-agent systems, Auction-based scheduling**

I. INTRODUCTION

Dial-a-ride paratransit systems play an instrumental role in providing equitable transportation services to special groups of the population, such as the elderly or handicapped. With a fare scheme comparable to that of a fixed-route transit, paratransit provides shared-ride, door-to-door service with flexible routes and schedules. The dial-a-ride problem (DARP) aims to optimally schedule and dispatch transit vehicles to satisfy requests for travel between pick-up and drop-off locations at specified times. A typical request in this context provides details on pick-up and drop-off locations, number of passengers, type of request, and optimal time windows within which it needs to be fulfilled. The scheduling component can be static, dynamic, or a combination of both. In the context of static DARP, all the requests and available vehicle fleet are known well in advance. In a dynamic DARP, the requests are serviced on an ongoing basis with an ability to increase the fleet size as needed. In the hybrid approach, reservations made in advance permit the construction of day-ahead (off-line) schedules, while the same day requests and other events such as trip cancellations, vehicle breakdowns etc. can be re-integrated into the current schedule (on-line schedule).

In the real world, DARPs have limited resources, making them suffer from over-subscription, where there is more demand than capacity. This leads to increasing the wait time and could even result in failure to fulfill a subset of transit requests within the specified time windows. This requires a careful trade-off between cost and time. A typical scenario in this context poses challenge on vehicle utilization and task allocation, for which an optimized destination-aware auction system is introduced based on dynamic and heterogeneous dial-a-ride problem (DH-DARP).

For the problem setting, we focus on the dynamic, heterogeneous DARP. The dynamic nature of the DH-DARP comes from multiple sources. For example, passenger requests are not all known in advance; they are submitted over time. In addition, traffic conditions can affect both pick-up and drop-off schedules so that the delay time is simulated in a stochastic fashion. Besides that, various car types (i.e. ambulance car and wheelchair carrier) are involved in a heterogeneous vehicle fleet in order to meet multiple requirements. Considering both function and capacity difference, requests are separately assigned to vehicles with consistent nature.

To make as many as passengers satisfied with the system under such a dynamic setting, we firstly propose our destination-aware strategy, which precisely divides requests into three priority categories: high priority goes for high-cost appointments, such as a doctor's appointment; medium priority deals with pre-organized valuable activities, where being on-time is preferable but not critical; and low priority is assigned to daily routines, such as grocery shopping. In this way, requests with lower level are given more relaxed time constraints, making room for requests with higher priority.

In addition to this strategy, this paper addresses the DH-DARP with an agent-based approach in terms of bid-computation, auction and request swapping. Each vehicle bids on a request based on how much it would cost the other passengers who have assigned to the vehicle. Every time an auction is completed, swapping is applied to improve the overall optimality of the whole system. The centrality and dispersion degree are also taken into consideration pursuing a comprehensive judgement for the network's performance.

The remainder of this paper is organized as follows. A review of related literature is provided in section II. Section III introduces a formal problem definition and a mathematical programming formulation of the problem. Section IV details the method we use, which encompasses system architecture as well as the proposed algorithms. Section V focuses on

[1]R. Martin is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA rebecca_martin@asu.edu

[2]Lan Wu is with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Shenzhen, Guangdong, China lanwu@link.cuhk.edu.cn

[3]Yu Wu is with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China wuyu1@shanghaitech.edu.cn

[4]I. Isukapati and S. Smith are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA (isaack,sfs)@cs.cmu.edu

simulation result and comparison with baseline method. Finally, a summary and suggestions for future research are provided in section VI.

II. RELATED WORK

Since proposed by Wilson in 1971 [2], the Dial-a-Ride Problem (DARP) has been studied extensively based on one-to-one Pickup and Delivery Problem with Time Windows (PDPTW). Generally, the scheduling component can be static, dynamic, or a combination of both, like stated in section I. In contrast to what happens in a static problem, the planning horizon of a dynamic DARP may be unbounded. Therefore, using gradually revealed information, it is crucial to specify a strategy for dynamic cases.

For static DARPs, recent research mainly focuses on exact, local neighborhood search and insertion heuristic techniques, together with a more accurate estimation model of fuel consumption [3] and service quality [4]. For example, exact methods such as branch-and-cut [5] have been developed based on mixed-integer programming for minimum cost. While considering complex constraints from capacity, duration, time window, to pairing and precedence, this method has a long execution time for large-scale problems. To overcome this weakness, heuristic approaches, GPU accelerated tabu search [6] and multi-atomic annealing heuristics [7] for instance, are presented with significantly improved efficiency and near-optimal solutions to various degrees.

Compared to static cases, far less work has been done for dynamic DARPs. One of the first studies on the Dynamic DARP was carried out by Psaraftis (1980) [8], who considered the single vehicle case. In order to minimize the average service period and reduce the dissatisfaction portion, an exact O($n^2 3^n$) algorithm was proposed to partially modify prior-generated routes. Then Horn (2002) [9] came up with strategical heuristics for large size system in software level. It periodically operates local search and considers practical manners like time window restriction, book cancellation, and future request anticipation. Moreover, aiming at more complex DARPs, Beaudry et al. (2010) have developed a two-phase algorithm, which considers requests' urgency level and various modes of transportation as substantial constraints [10].

The scheduling of the DH-DARP can be abstracted as a dynamic multi-agent traveling salesman problem (TSP). In a TSP, the vehicle has to find a schedule and follow it to travel all the required locations without repeated visit and violating any constraint. The traveling salesman problem is a well-known NP-hard problem and many algorithms have been proposed for solving it under different constraints. The most popular method uses a greedy strategy with an optimization algorithm to improve the schedule with respect to a particular heuristic. Other less common methods utilize an auction-based framework. In current literature, most of these methods make overly simplifying assumptions, such as that each agent can only carry out one job at a time, otherwise, the schedule can only be computed off-line, which is not dynamic as we want. Therefore, mainly lacking in the prior work is a multi-agent scheduling system that reasons in an optimized and stable fashion about vehicle utilization and role adoption while maintaining the ability to respond dynamically and efficiently to unexpected requests under large size constraints. In this paper, we demonstrate a continuous, agent-based scheduling of trip requests with a market approach [11] having its primary advantage in opportunistically optimizing resource utilization. We will consider using a bidding system to construct dynamic schedules, optimized for time and capacity, while accounting for the different accommodations needed for wheel-chair bound and ambulatory passengers. Note that since each vehicle is modeled as self-interested agent, the overall cost is minimized thus benefiting the whole market.

III. PROBLEM FORMULATION

In the dynamic DARP, the requests r are sent continuously along with the operation of the vehicle fleet V.

In the vehicle fleet V, there are n working vehicles, with each one named $v_i, i \in [1, n]$. In each vehicle v_i, there is capacity of α_{v_i} for wheel-chair passengers and capacity of β_{v_i} for ambulatory passengers. Each vehicle also has its accepted request list R_{v_i} and current schedule S_{v_i} where each schedule s in it is a schedule dictionary, with key-value pairs of trip ID j_s from the corresponding request, scheduled location l_s, scheduled time t_s, and passenger priority p_s in them, which depends on the passengers' destination. We set all the vehicles to be identical on velocity and define the expected traveling time between any two locations l_1 and l_2 for any v_i to be $E(l_1, l_2)$. This value of it is computed by a third-party API, which is introduced more in section IV.

For any request r, j_r is the unique trip ID, and the following information is contained. $l_r^{(p)}$ and $l_r^{(d)}$ represent the locations for pick up and drop off the passenger. $T_r^{(p)}$ and $T_r^{(d)}$ stand for the allowed time windows (closed intervals of time) for pick-up and drop-off time. x_r is the number of wheel-chair passengers and y_r is the number of ambulatory passengers. p_r is the priority of the passengers.

For any certain $i \in [1, n]$, $\forall s \in S_{v_i}$ and $\forall r \in R_{v_i}$, it should be satisfied that:

$$(j_s = j_r) \Rightarrow (l_s = l_r^{(p)} \vee l_s = l_r^{(d)}) \tag{1}$$

$$(j_s = j_r) \Rightarrow (p_s = p_r) \tag{2}$$

$$(j_s = j_r \wedge l_s = l_r^{(p)}) \Rightarrow (t_s \in T_r^{(p)}) \tag{3}$$

$$(j_s = j_r \wedge l_s = l_r^{(d)}) \Rightarrow (t_s \in T_r^{(d)}) \tag{4}$$

For any $i \in [n]$,

$$\alpha_{v_i} \geq \sum_{r \in R_{v_i}} x_r \tag{5}$$

$$\beta_{v_i} \geq \sum_{r \in R_{v_i}} y_r \tag{6}$$

For any s_j and s_k in any certain S_{v_i},

$$|t_{s_j} - t_{s_k}| \geq E(l_{s_j}, l_{s_k}) \tag{7}$$

Among all above equations, equation (1, 2, 3, 4) respectively show the correspondences of the locations, priorities, scheduled times between a schedule dictionary and a request. Equation (5) and (6) constrain the capacity demands, and equation (7) limits the time differences between any two scheduled locations according to the expected time duration.

Suppose for any s the allowed time window for l_s is T_s, given the constraints above, the optimization object is by changing S_{v_i}'s to achieve:

$$\min \sum_{i \in [n]} \sum_{s \in S_{v_i}} disutil((t_s - \min(T_s)), p_s) \tag{8}$$

where $disutil$ is defined as:

$$disutil(\Delta t, p) = \begin{cases} \log_2 \Delta t & \text{p=1} \\ \Delta t & \text{p=2} \\ (\Delta t)^2 & \text{p=3} \end{cases} \tag{9}$$

The intention of disutility function $disutil$ is to give the requests with different priorities different non-linear weights, which attaches high priority requests apparent importance.

In sum, given all the constraints, we want to minimize the gap between all the scheduled time and negotiated time, for all the vehicles, with non-linear mappings decided by passengers' priorities as weights.

IV. METHODS

A. External Data and Tools

In this system, we used Open Route Service to calculate the distances and travel times between two locations. Our simulator is based on [12]. Historical paratransit data from Pittsburgh's ACCESS paratransit system was used for testing. We started out with a set of 902 paratransit ride requests for one day. For this research, these requests were augmented with priority tags 1, 2, or 3, uniformly at random.

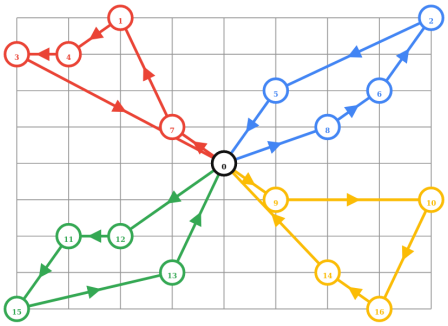


Fig. 1. A sample solution to a VRP using Google OR-Tools.

We used OR-Tools from Google to compute optimal schedule for one vehicle. The scheduling problem for a single vehicle was modeled as a TSP with time windows, which can be solved by adding the time matrix and time windows. The time matrix stores the travel times between locations and the time windows limit the allowed visiting time of the locations. Because the original version of the tool is for solving vehicle routing problems (VRPs), which are the multi-agent version of TSPs, we set the vehicle number equal to one. It is worth noting that the solver only deals with static scheduling problems, and we invoke it repeatedly to fit our dynamic settings. Figure 1 from OR-Tools website shows an example of the solution to a VRP using OR-Tools with four vehicles.

B. Scheduling Algorithm

The total procedure of assigning a new request to one of the vehicles works like Algorithm 1. Each vehicle has the equal right to bid for the request, and the one which expects to be have the least cost on its previous schedule wins the bid. When a vehicle is bidding, the virtual updated schedule is computed using OR-Tools. The winner of the bid accepts the request and replaces its schedule with the updated one.

Algorithm 1 Check and assign new requests to the vehicles.

Input: The new request, r; The global vehicle list, V;

```
c_min ← ∞;
v_opt ← None;
for all v in V do
    Compute cost c supposing v accepts r;
    if c < c_min then
        c_min ← c;
        v_opt ← v;
    end if
end for
Assign r to v_opt and update v_opt's schedule;
```

The detailed bidding procedure is showed as algorithm 2. If any conflict happens, the vehicle has to abandon the bid. The updated schedule is compared with the old schedule of the vehicle, and any additional delay on any previously scheduled stop will accumulate the cost. In addition, the delay of a passenger of a previously accepted request with high priority will result in a higher cost. The vehicle which has the lowest expected cost for the new request wins the bid and accepts the request.

Another method applied is swapping some requests among some vehicles when swapping can optimize the global schedule, shown as algorithm 3. The swap check happens in a stable frequency, ensuring that each swap helps reduce the total cost and improve the dispersion and equity of the distribution of the vehicles. Any request with too short remaining time till pickup cannot be swapped.

It should be noted that the dispersion and equity values are important factors calculated in the process of swapping. The equity value considers how many vehicles are there within the specified distance d with respect to given popular locations P_i. In other words, we anticipate that another passenger will dial a ride, choosing one of those popular locations as his pick-up address. Shortly thereafter, his car arrives with an expected travel period $\hat{t}$. Then, the equity value measures how accessible each popular location will be at time $\hat{t}$, which gives a weighted average taking the relative popular index P_i^{index} into consideration.

Algorithm 2 Compute cost for a vehicle given a new request.

Input: The new request, r; The vehicle instance, v;
Output: The cost, c;

if v has no capacity for the passenger in r **then return** ∞;
end if
Add r to v's request list to get a new virtual request list R^*;
From R^* and v compute parameter set $Param$ needed for OR-Tools;
Call OR-Tools API with $Param$ to get new schedule S^*;
if S^* == $None$ **then return** ∞;
end if
$c \leftarrow 0$;
for all schedule dictionary s in v's old schedule S **do**
    $t \leftarrow$ corresponding scheduled time of s in S;
    $t^* \leftarrow$ corresponding scheduled time of s in S^*;
    $delay \leftarrow t^* - t$;
    $p \leftarrow$ corresponding passenger priority of s;
    $c \leftarrow c + disutil(delay, p)$; //$disutil$ is shown in section III.
end for
return c;

Similar to the equity value, the dispersion value employed here measures the spread of the vehicle fleet over the city according to the distance of each vehicle at some future anticipated time $\hat{t}$. At that point, the average straight-line distance between each possible pair of vehicles is calculated as below:

$$d_v^{t_i} = (\frac{2!(|v|-2)!}{|V|!}) \sum_{i \in V} \sum_{j \in (V: j \neq i)} d_{ij} \tag{10}$$

$$t_i \Rightarrow N(\hat{t}, \sigma^2) \tag{11}$$

$$d_v = \sum d_v^{t_i} P(t_i) \tag{12}$$

V. RESULTS

To test these schedules and incorporate some of the uncertainty present in carrying out the schedules, we used the statistical simulator from [12]. In this simulator, the paratransit schedules were randomized by sampling from two uniform distributions. The first distribution determined whether the vehicle would arrive at its stop early, on-time, or late. If the vehicle was not on-time, the second distribution determined how early or late the vehicle was going to be, with a maximum of five minutes between the scheduled time and the randomized time. However, to preserve the spatio-temporal aspects of each trip, the randomized time between two stops was not allowed to be less than the minimum travel time calculated by the Open Route Service API.

The baseline schedule was created statically using a greedy scheduling algorithm to construct an initial schedule. This initial schedule was then given as input to Generalized Task Swap [1], a local search algorithm, which optimized the schedule and made sure that all requests were scheduled within the given time constraints, if they had not managed to fit into the initial schedule.

For our experiments, we tested to see how much the order the requests were received in affected the performance. For this, we had requests submitted either exactly 30 min or randomly between 30 and 45 min before their requested pickup time. We ran ten experiments for each condition, while varying the random seed. These experiments were then compared to the static optimized schedule based on ride duration. We computed the improvement in the sum, mean, and median of all of the ride durations for the entire schedule. These results are shown in Table I below. We also plotted the CDFs of the distribution of ride times to visualize the improvement offered by the auction system. These CDFs have been cropped at the 95th percentile to eliminate outliers. The graphs are shown in Figures 2, 3, and 4.

From this data, we can see that the median ride durations improved in every case for all priorities. The mean ride duration also improved significantly for all requests, regardless of the order of request arrivals. This means that the vast majority, if not all, of the requests saw a decrease in ride duration, with the longer rides receiving more benefit. This improvement in ride times is because the auction system has less interleaving of pickup and droppoffs; that is, it keeps the ride time for each request short by making each passenger's trip as direct as possible, while minimizing overall system cost.

VI. CONCLUSION

In this paper, we created an auction-based scheduling algorithm for paratransit systems. This dynamic system handled same-day ride requests and considered request priorities based on the destination. Our scheduling algorithm was tested in a Python simulator and compared against a static optimization algorithm [1]. Overall, the auction-based system performed well, with both mean and median ride durations decreasing by at least 35%. In future work, we will test our request swapping algorithm as an additional layer in this auction-based system.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774, the Chinese University of Hong Kong at Shenzhen, and ShanghaiTech University. The authors thank Rachel Burcin, John Dolan, and all of Carnegie Mellon University's Robotics Institute Summer Scholars (RISS) program.

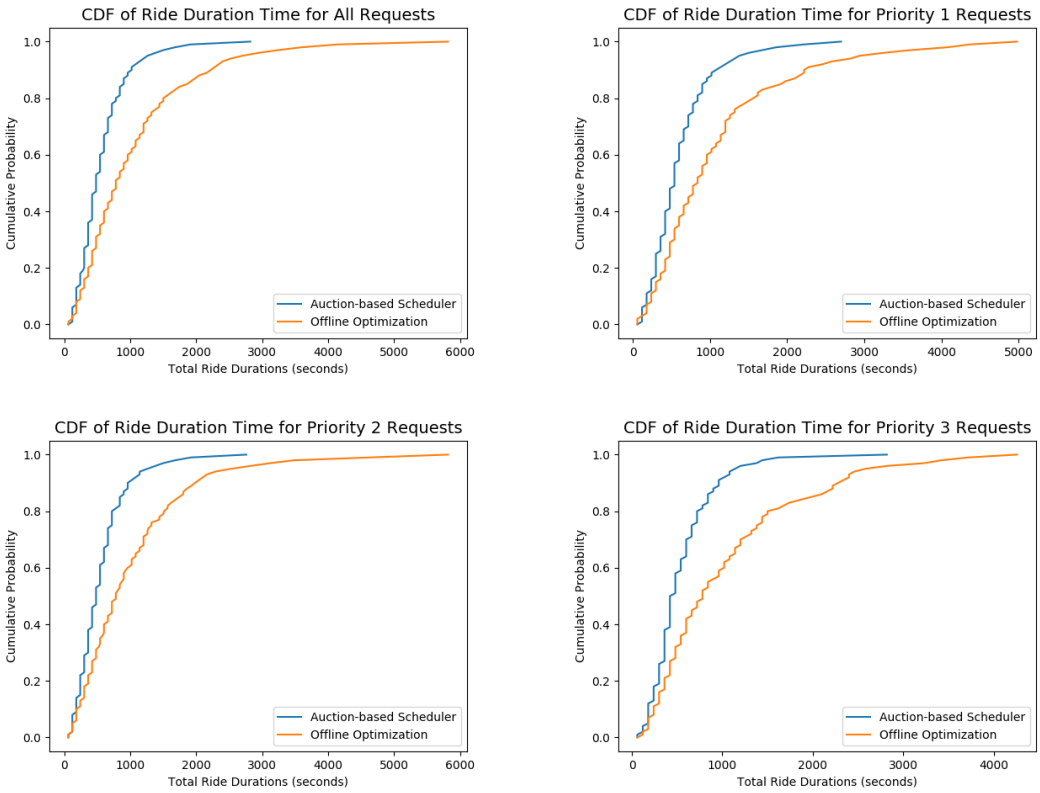


Fig. 2. CDFs of the scheduled ride duration times in seconds, when request arrival is fixed at 30 minutes. Top row: All requests, Priority 1 requests; bottom row: Priority 2 requests, Priority 3 requests.

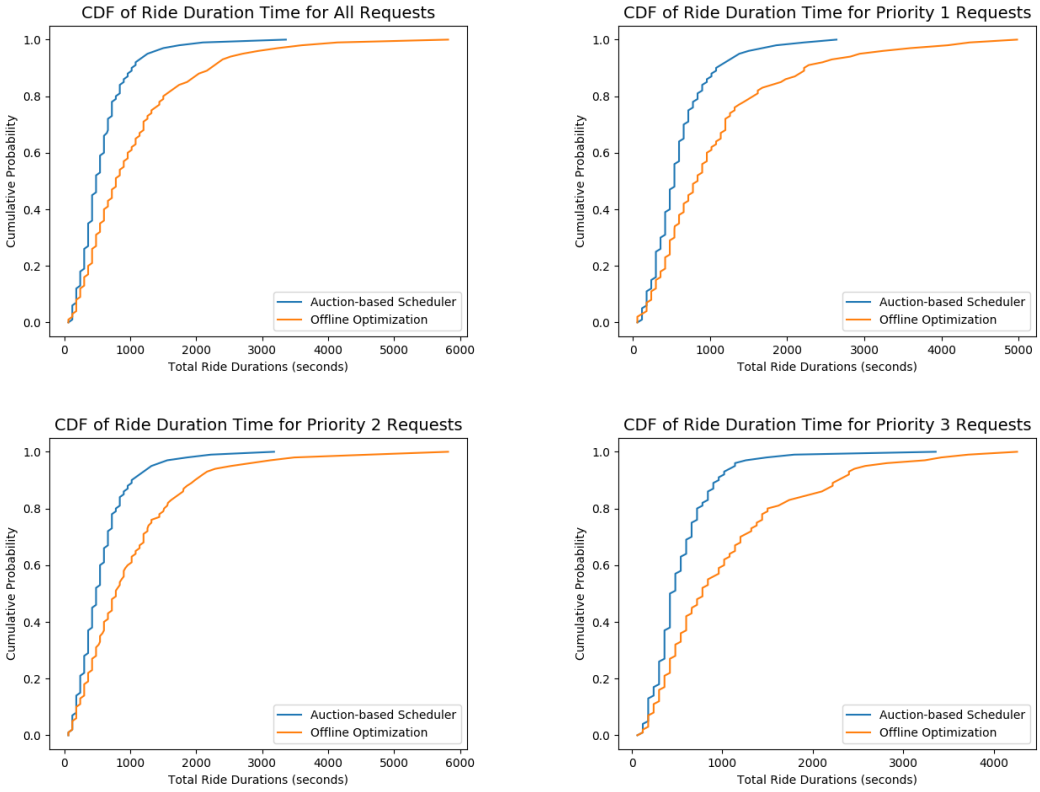


Fig. 3. CDFs of the scheduled ride duration times in seconds, when request arrival is randomized between 30 and 45 minutes. Top row: All requests, Priority 1 requests; bottom row: Priority 2 requests, Priority 3 requests.

Algorithm 3 Check and swap requests.

Input: The global vehicle list, V; Allowable minimum remaining time to begin the serving of requests, T_{min};

for all v in V **do**
  $S \leftarrow$ the schedule of v;
  for all schedule dictionary s in S **do**
    $req \leftarrow$ corresponding request of s;
    $t_p \leftarrow$ scheduled pick up time for req;
    if t_p - current time $\geq T_{min}$ **then**
      $c' \leftarrow$ the profit supposing v deletes req;
      $S' \leftarrow$ the supposed new schedule of v;
      $eval' \leftarrow$ dispersion value plus equity value of the distribution of V with S' considered;
      $v_{opt} \leftarrow None$;
      $eval_{max} \leftarrow 0$;
      for all v^* in $V - v$ **do**
        $c \leftarrow$ the cost supposing v^* accepts req;
        $S^* \leftarrow$ the supposed new schedule of v^*;
        if $c' > c$ **then**
          $eval^* \leftarrow$ dispersion value plus equity value of the distribution of V with S' and S^* considered;
          if $eval > eval_{max}$ **then**
            $eval_{max} \leftarrow eval$;
            $v_{opt} \leftarrow v^*$;
          end if
        end if
      end for
      if $v_{opt} \neq None$ **then**
        Delete req from v and update v's schedule;
        Assign req to v_{opt} and update v_{opt}'s schedule;
        return
      end if
    end if
  end for
end for

TABLE I

ARRIVAL REQUEST FIXED AT 30 MINUTES

Scenario	Sum Total Duration (sec)			Mean Duration (sec)			Median Duration (sec)		
	Auction-Based	Offline Optimization	% Improvement	Auction-Based	Offline Optimization	% Improvement	Auction-Based	Offline Optimization	% Improvement
Overall trips	507084	935460	45.79%	562.18	1037.10	45.79%	480	780	38.46%
Priority 1	175296	310260	43.50%	602.39	1066.19	43.50%	528	840	37.14%
Priority 2	165468	303360	45.45%	551.56	1011.20	45.45%	480	780	38.46%
Priority 3	166320	321840	48.32%	534.79	1034.86	48.32%	438	780	43.85%

TABLE II

ARRIVAL REQUESTS RANDOMIZED BETWEEN 30 AND 45 MINUTES

Scenario	Sum Total Duration			Mean Duration			Median Duration		
	Auction-Based	Offline Optimization	% Improvement	Auction-Based	Offline Optimization	% Improvement	Auction-Based	Offline Optimization	% Improvement
Overall trips	515268 sec	935460 sec	44.92%	571.25 sec	1037.10 sec	44.92%	480 sec	780 sec	38.46%
Priority 1	177282 sec	310260 sec	42.86%	609.22 sec	1066.19 sec	42.86%	540 sec	840 sec	35.71%
Priority 2	170976 sec	303360 sec	43.64%	569.92 sec	1011.20 sec	43.64%	483 sec	780 sec	38.08%
Priority 3	167010 sec	321840 sec	48.11%	537.01 sec	1034.86 sec	48.11%	438 sec	780 sec	43.85%

TABLE III

UNCERTAINTY IN PICKUP/DROPOFF TIMES

Scenario	Sum Total Duration			Mean Duration			Median Duration		
	Auction-Based	Offline Optimization	% Improvement	Auction-Based	Offline Optimization	% Improvement	Auction-Based	Offline Optimization	% Improvement
Overall trips	584417 sec	947892 sec	38.35%	647.91 sec	1050.88 sec	38.35%	565.4 sec	801.5 sec	29.46%
Priority 1	200920 sec	314189 sec	36.05%	690.45 sec	1079.69 sec	36.05%	597.9 sec	822.0 sec	27.26%
Priority 2	190632 sec	307295 sec	37.96%	635.44 sec	1024.32 sec	37.96%	565.9 sec	782.2 sec	27.65%
Priority 3	192865 sec	326407 sec	40.91%	620.15 sec	1049.54 sec	40.91%	544.8 sec	796.1 sec	31.57%

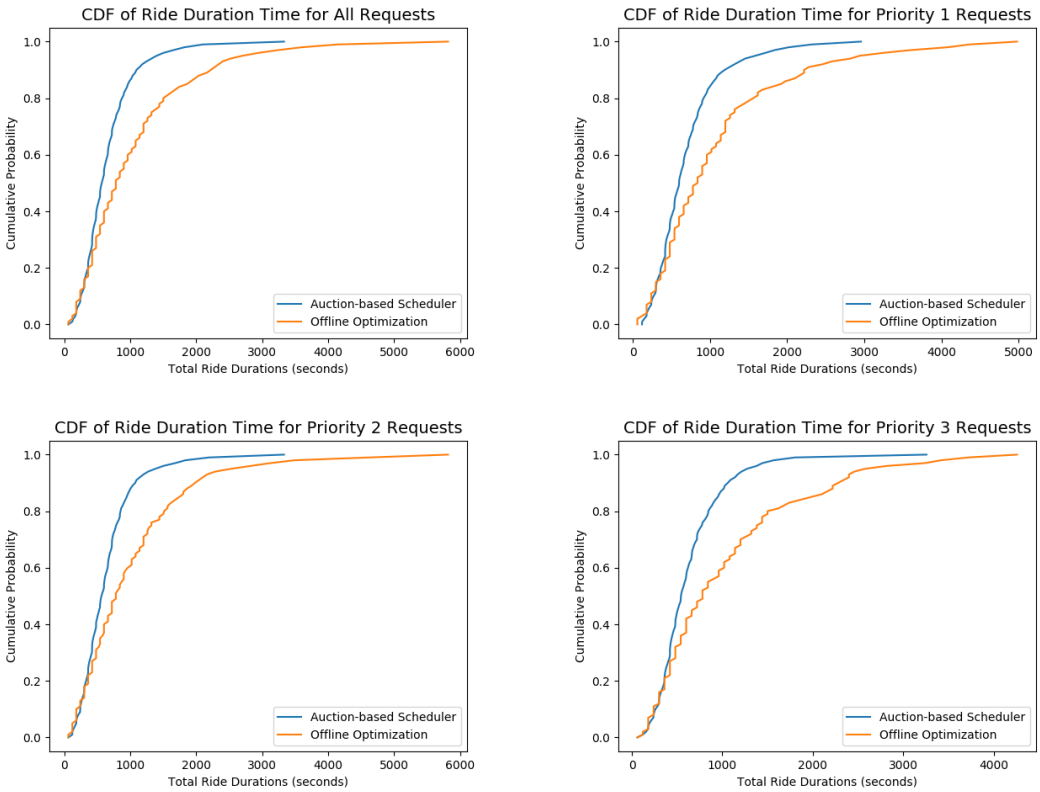


Fig. 4. CDFs of the simulated ride duration times in seconds, when request arrival is fixed at 30 minutes. Top row: All requests, Priority 1 requests; bottom row: Priority 2 requests, Priority 3 requests.

REFERENCES

[1] Z. B. Rubinstein, S. F. Smith, and L. Barbulescu, "Incremental management of oversubscribed vehicle schedules in dynamic dial-a-ride problems," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[2] N. H. Wilson, J. M. Sussman, H.-K. Wong, and T. Higonnet, *Scheduling algorithms for a dial-a-ride system*. Massachusetts Institute of Technology. Urban Systems Laboratory, 1971.

[3] W. Knörr, S. Seum, M. Schmied, F. Kutzner, and R. Anthes, "Ecological transport information tool for worldwide transports," in *Methodology and data update*. IFEU Heidelberg, Oko-Institut, 2011.

[4] J. Paquette, F. Bellavance, J.-F. Cordeau, and G. Laporte, "Measuring quality of service in dial-a-ride operations: the case of a canadian city," *Transportation*, vol. 39, no. 3, pp. 539–564, 2012.

[5] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[6] R. R. Pandi, S. G. Ho, S. C. Nagavarapu, T. Tripathy, and J. Dauwels, "Gpu-accelerated tabu search algorithm for dial-a-ride problem," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2519–2524.

[7] S. G. Ho, R. R. Pandi, S. C. Nagavarapu, and J. Dauwels, "Multi-atomic annealing heuristic for the dial-a-ride problem," in *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. IEEE, 2018, pp. 268–273.

[8] H. N. Psaraftis, "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, vol. 14, no. 2, pp. 130–154, 1980.

[9] M. E. Horn, "Fleet scheduling and dispatching for demand-responsive passenger services," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 1, pp. 35–63, 2002.

[10] A. Beaudry, G. Laporte, T. Melo, and S. Nickel, "Dynamic transportation of patients in hospitals," *OR spectrum*, vol. 32, no. 1, pp. 77–107, 2010.

[11] D. Goldberg, V. Cicirello, M. B. Dias, R. Simmons, S. Smith, and A. Stentz, "Market-based multi-robot planning in a distributed layered architecture," in *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings of the 2003 International Workshop on Multi-Robot Systems*, vol. 2, 2003, pp. 27–38.

[12] R. Martin, I. Isukapati, S. Smith, and Z. Rubinstein, "Using fixed route transit to improve paratransit service quality," unpublished.

Subtask Classification with Eye Gaze for Teleoperated Tasks

Kathleen Medill,[1] Maggie Collier[2] and Henny Admoni[2]

***Abstract*— Teleoperated robot arms can greatly improve the quality of life for individuals with disabilities by assisting them with everyday tasks. Although these assistive arms are effective at some simple one-step tasks, they are difficult to use for more complex multi-step activities. To address this challenge of teleoperation, researchers have begun experimenting with shared autonomy, which integrates human control and autonomous robot control to make assistive robots easier to teleoperate. To date, however, shared autonomy has only been used successfully with simple one-step tasks. To scale shared autonomy to multi-step activities, it is important to identify the current subtask the user is attempting. This work focuses on classifying the immediate subtask within a multi-step activity by incorporating features from the user's eye gaze into a compact generalized non-local (CGNL) network [1], a state-of-the-art action recognition approach. In a pilot study, we collected eye gaze data as a user teleoperated a robot arm to complete a food preparation task consisting of four subtasks. Incorporating eye gaze into the CGNL model boosted the classification accuracy of subtasks by 9.52% for subtask sequences that followed an uncommon order and had accurate eye gaze data. This results indicates that including eye gaze can potentially make CGNL more robust to multi-step tasks without a fixed subtask order.**

***Index Terms*— Human-robot interaction, eye gaze, action recognition**

I. INTRODUCTION

Assistive technology, such as wheelchair-mounted robot arms, has been shown to greatly improve the quality of life for individuals with disabilities. These devices allow users to live more independent lives by assisting them with simple everyday tasks, such as picking up a dropped item or taking a drink. Although assistive arms are effective at many simple one-step tasks, they are difficult to use for more complex multi-step activities. Typically, these robots have a high number of degrees-of-freedom (DOF) but are teleoperated via a low DOF interface, such as a joystick or sip-n-puff. As a result, users must manually switch between modes to utilize the different DOFs of the robot, thereby increasing the steps and time required to complete a task. As a result, multi-step activities of daily living such as food preparation or dressing, which are simple and quick when performed with one's hands, are much longer and complex when completed with a teleoperated assistive arm, often causing a frustrating user experience.

To address this problem, researchers have investigated ways to partially automate the controls of teleoperated robots via shared autonomy [2], [3]. Shared autonomy integrates human control (from an input device like a joystick) and autonomous robot control to alleviate the burden on the user caused by manually switching between modes to access all the necessary robot DOFs. Thus, shared autonomy has the potential to create a system that is easier to teleoperate and more user friendly. To date, however, shared autonomy has only been shown to be successful with simple one-step tasks. Shared autonomy could make many multi-step activities of daily living significantly easier or, in some cases, possible to be completed via teleoperation. Moreover, enhancing shared autonomy so that it can be applied to multi-step activities would greatly improve the usefulness of teleoperated robotic arms and further increase the independence of their users.

To achieve scaling shared autonomy to multi-step tasks, we can start by giving shared autonomy some notion of which subtask within a multi-step activity the user is attempting to accomplish. The user's eye gaze could be a valuable signal for classifying the immediate subtask. Many psychology and hand-eye coordination studies have shown that individuals mainly look at task-relevant objects during object manipulation tasks, therefore indicating their future goals with their eye gaze [4], [5]. Additionally, eye gaze is an implicit natural behavior. Therefore, using eye gaze as an input to a shared-autonomy system could provide valuable information about the user's goals without putting additional strain on the user.

Action recognition in computer vision is a promising avenue for subtask classification. Additionally, including eye gaze information in action recognition has proven to be a useful strategy for subtask classification in the context of multi-step activities demonstrated to robots by human teachers [6], and many strategies in egocentric action recognition ([7]–[9] to name a few) often predict the user's gaze and use this prediction to better inform action classification . While this field is promising, to our knowledge, none of the state-of-the-art action recognition approaches have been evaluated in the context of teleoperated actions. In this paper, we propose incorporating features from the user's eye gaze into a state-of-the-art action recognition model to improve the classification of subtasks completed using a teleoperated assistive arm. More specifically, we investigated the performance of a compact generalized non-local (CGNL) network [1] with two different gaze featurization approaches. By including a trial in our analysis with lower confidence gaze readings we demonstrate the necessity of high confidence gaze readings and discuss a potential metric for determining which gaze data are accurate enough to be useful for classification. As our major finding, we demonstrate that gaze data can improve

[1]Kathleen Medill is with the Electrical and Computer Engineering Department, U.S Air Force Academy, Colorado Springs, Colorado C21Kathleen.Medill@afacademy.af.edu

[2]Maggie Collier and Henny Admoni are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsyvania {macollie,hadmoni}@andrew.cmu.edu

classification on trials containing subtask sequences that are underrepresented in the dataset, a situation that simulates a real-world difference in user preferences and teleoperation strategies.

II. Related Work

A. Eye Gaze in Object Manipulation Tasks

Psychologists have long known that eye gaze is an important part of human-to-human interaction, particularly for communicating about objects and locations in the environment. Prior psychological research has studied the gaze behavior of people performing certain manipulation tasks with their own hands, such as moving objects around obstacles [10] or making tea [4]. These studies show that eye gaze follows the objects involved in the task [11] and that eye gaze precedes hand motion [10], [12]. When people act to manipulate objects, their gaze typically reaches the target before they even begin moving their hands [12], and shifts to the next target before their hands reach the current target [10]. More recent research has studied eye gaze in the context of teleoperated object manipulation tasks. Aronson, et al. conducted experiments using an eye tracker to record eye gaze during a human-robot shared manipulation activity, both with and without shared autonomy assistance [13]. The study identified novel patterns of gaze behavior that were different from previous findings about eye gaze in human-only activity. The results suggested that a user's natural eye gaze contain features that could be used to enhance teleoperated manipulation tasks.

B. Action Recognition in Computer Vision

Action recognition in computer vision is an active area of research that has experienced much recent success, and several approaches incorporate eye gaze information to better inform action recognition ([7]–[9] to name a few). Of the gaze-based approaches, many include a gaze prediction step and add this prediction into an action recognition module. These strategies are benchmarked on datasets of people completing manipulation tasks with their hands, and little to no work has attempted to evaluate state-of-the-art approaches in different manipulation contexts. One notable work uses two state-of-the-art approaches for subtask classification in the domain of robot learning from demonstration [6]. In this work, Saran, et al. studied eye gaze patterns of humans teaching tasks to robots via video and kinesthetic demonstrations [6]. Although not explored in the context of teleoperation, this study demonstrated that the teacher's eye gaze enhanced subtask classification for multi-step tasks up to 6%. To investigate subtask classification, Saran, et al., modified two promising action recognition approaches from computer vision [1], [14] by adding the teacher's raw gaze positions before the final layer of the model. One of the approaches, from Wang, et al., introduced the concept of including a non-local network (NL) as a generic component for capturing long-range dependencies in a deep neural network [14]. Yue, et al., expanded this approach with compact generalized non-local network (CGNL) [1], demonstrating that CGNL is able to model more complex interactions among features than NL. To our knowledge, none of the state-of-the-art action recognition approaches have been evaluated in the context of teleoperation or with gaze features other than raw gaze positions. For the current study, we chose to use CGNL due to Saran, et. al's success incorporating gaze and applying it to kinesthetic teaching videos. In future work, we will evaluate some of the action recognition approaches that incorporate raw gaze predictions on our data set and investigate incorporating different gaze features.

III. Methods

In this study, we sought to improve the classification of subtasks in a multi-step food preparation activity completed with a teleoperated robot arm. Like Saran, et al [6], we used a CGNL network [1] as our baseline approach, and we incorporated eye gaze into CGNL in two different ways.

A. Task Design and Data Collection

Previous work from our lab included a pilot study to obtain the data we used in our experiments. In the pilot study, the secondary author recorded eye gaze data while teleoperating an assistive arm to perform a multi-step food preparation task six times. In each trial, the user teleoperated a Kinova MICO robotic arm to accomplish the food preparation activity while wearing a pair of Pupil Labs gaze tracking glasses [15], [16]. The Pupil Labs glasses have three cameras–a world camera capturing the wearer's egocentric view and two cameras positioned on each of the wearer's eyes. After the tracker is calibrated, the accompanying software maps the positions of the wearer's pupils to a position estimate in the egocentric camera, along with a confidence score for that estimate.

The six trials contained the same multi-step task: serving food on a plate on a placemat. We defined four subtasks for classification. We hand annotated the end of each subtask using the following protocol:

- Grasp Spoon - when the fingers had closed round the tool holding the spoon
- Scoop Food - when the spoon contained food and was no longer inside the food container
- Dump Food - when the food first touched the plate
- Push Plate - when the plate was on the place mat and was no longer moving

Examples of the user's view while completing each subtask are shown in Fig. 1. Trials 1, 2, 4, and 5 followed a predominant subtask order of push plate, grasp spoon, scoop food, dump food. Trials 3 and 6 followed an alternative subtask order: grasp spoon, scoop food, dump food, push plate. The purpose of changing the order of the subtasks was to represent possible differences in user preference and task completion strategy: subtask order differences were observed in a similar pilot study conducted in the lab that involved multiple participants.

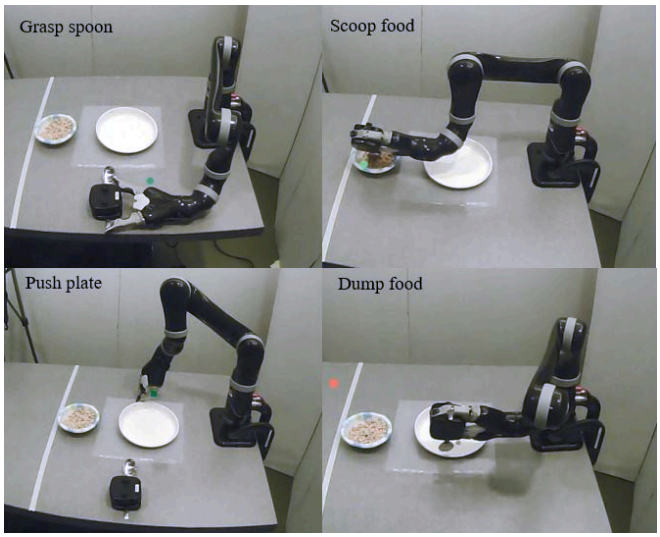


Fig. 1. Still frames showing each of the four subtasks being completed. The green and red dots are at the location of the user's gaze. The green dots represent higher confidence gaze readings than the red dot.

B. Subtask Classification

All approaches for our study use or modify the 2D implementation of CGNL outlined in [1]. Thus, we use CGNL like an image classifier, where each input consists of a single frame from the egocentric video and is associated with a single subask label. Action recognition takes a sequence of frames from a video clip as a single sample, thereby requiring a significantly larger dataset. Thus, using CGNL like an image classifier enabled us to evaluate on our small pilot dataset. In future work, we will expand our CGNL implementation to accept sequences of frames sampled from video, as traditionally done in action recognition approaches, and collect significantly more trials to build a large enough dataset for action recognition.

For each approach and before incorporating gaze data, we sampled frames of the worldview video at 15 fps from the six pilot study trials. In an additional step, we excluded any frames that had a corresponding gaze position estimate with a confidence score lower than 0.5. The resulting frames were used and/or modified for all three approaches.

For all experiments, we used a ResNet-50 architecture with one CGNL block included and used the same image augmentations and training parameters defined in the CGNL paper [1]. To establish a baseline, we excluded all gaze data and only used the frames we sampled from the egocentric camera.

We then incorporated eye gaze information from the same six trials into the same CGNL model in two different ways. The first approach, the raw gaze approach, is taken from Saran, et al. [6]. In the raw gaze approach, the normalized gaze position estimate for each input frame was added into the final layer of the model.

For the overlay gaze approach, we added the eye gaze information directly to the input frames. We accomplished this by overlaying a uniquely patterned dot at the estimated location of the user's gaze for each sampled frame. Fig. 2 shows examples of frames with the gaze dot overlaid. The gaze dot has a unique pattern to prevent the dot from being indistinguishable after the input frames undergo augmentations (the standard image augmentations done in image classification and action recognition).

Fig. 2. Example of sampled input frame with dot overlaid at position of eye gaze.

IV. Results and Discussion

To assess each approach, we conducted 6-fold cross validations for the baseline model, the raw gaze model, and the overlay gaze model. (In each fold, the model was trained on samples from five trials and then tested on samples from the remaining trial.) The accuracy averaged across the 6 folds are as follows:

- Baseline: 91.58%
- Raw gaze: 89.29%
- Gaze overlay: 89.99%

The results shown in Fig. 3 demonstrate that adding eye gaze (as raw positions or overlaid onto the input frame) only slightly changed the accuracy of the baseline model for the predominant order of subtasks in Trials 1, 2, 4, and 5. For the other order of subtasks in Trial 3, however, the raw gaze approach had a 4.17% improvement and the gaze overlay approach had a 9.52% improvement. For Trial 6, adding raw gaze and overlaid gaze resulted in a decline in accuracy from the baseline model, which we found was caused by a high percentage of samples having low-confidence gaze estimates.

The trials that followed the predominant subtask order–Trials 1, 2, 4, and 5–all had very similar results. On these trials, the raw gaze model achieved accuracies between 0.29% and 1.44% higher than the baseline model. However, the gaze overlay model showed accuracies between 0.08%

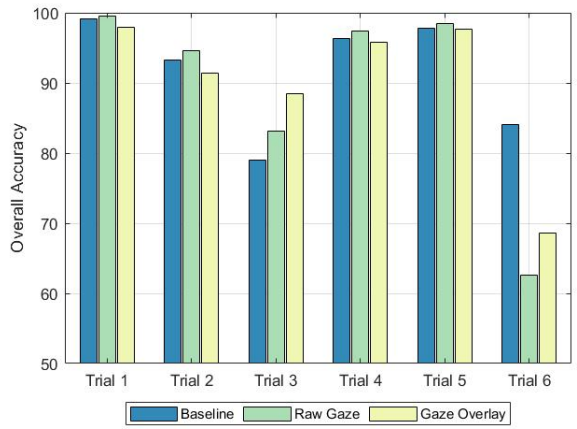


Fig. 3. Comparison of sub-task classification accuracies by trial and methods. Although Trials 1, 2, 4 and 5 were barely affected by the addition of eye gaze, Trial 3 had a 4.17% improvement for the raw gaze approach and a 9.52% improvement for the gaze overlay approach. The decrease in accuracy in Trial 6 can be attributed to the low confidence scores of its gaze position estimates.

and 1.75% lower than the baseline. The results indicate that our two approaches of adding eye gaze had a minimal effect on the overall accuracy for trials with the predominant ordering of the four subtasks. This result was not entirely unexpected, however, because the baseline approach achieved high accuracies on Trials 1, 2, 4, and 5 (84.05% to 99.18%).

The raw gaze model improved the accuracy for every trial except Trial 6. This result was expected because of poor quality gaze estimates in Trial 6, likely caused by an error during the calibration of the Pupil glasses. Only 50.09% of samples in Trial 6 had a confidence score greater than 0.90, compared to 76.39% to 90.44% of samples for the other five trials. Nevertheless, we still included Trial 6 to gain a better understanding of the weaknesses of our approaches. Including Trial 6 allows us to assess how low confidence gaze data can adversely affect the accuracy of otherwise high performing subtask classification approaches. We also observed that the percentage of samples above a certain confidence score could be used to define exclusion criteria or a way to identify an appropriate threshold for confidence scores.

The accuracy of Trial 6 was significantly decreased from 84.05% for the baseline model to 62.62% for the raw gaze model, but the gaze overlay model showed less of a decrease with 68.57% accuracy. In the future, we plan to increase the confidence threshold to 0.85 or 0.90, based on the large percentage of samples found above these threshold in Trials 1 through 5, to ensure better quality gaze data. Using higher confidence gaze data, we suspect that we would see similar gains in accuracy for Trial 6 as we saw for Trial 3, which had the same subtask order.

Trial 3 showed the greatest improvement from the baseline model to the raw gaze model with an improvement in accuracy of 4.17%. Accuracy gains were even greater in the overlay gaze model, which produced an improvement in accuracy over the baseline model of 9.52%. Overall, Trial 3 went from an accuracy rate of 78.95% for the baseline model to 88.47% for the overlay gaze model. This was an especially encouraging result because Trial 3 had the worst accuracy results of any trial for the baseline model. We suspect that Trial 3 was the least accurate using the baseline model because it was one of only two trials in which the subtasks were completed in the non-predominant order. Therefore, there was only one trial (Trial 6) completed in the same order included the training set for that fold.

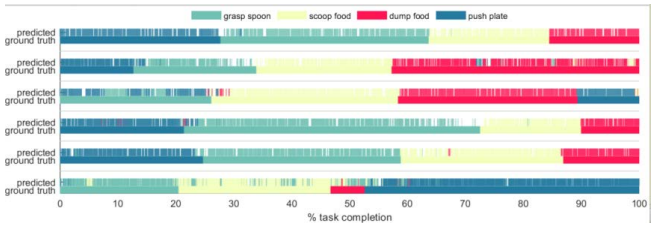


Fig. 4. Qualitative view of baseline model predictions. Trial 1 is shown at the top and Trial 6 at the bottom.

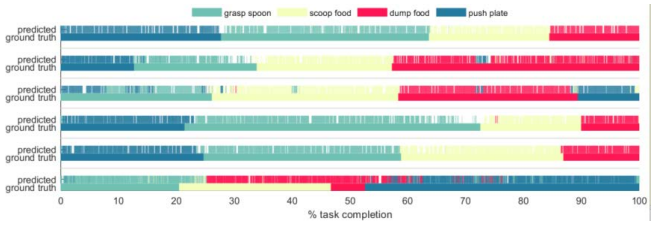


Fig. 5. Qualitative view of raw gaze model predictions. Trial 1 is shown at the top and Trial 6 at the bottom.

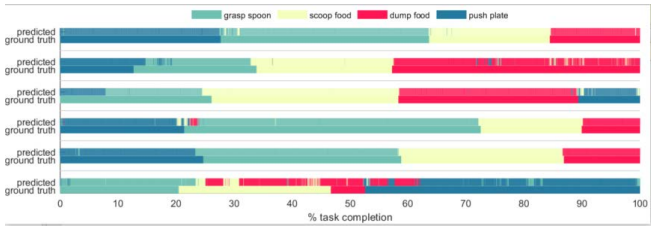


Fig. 6. Qualitative view of gaze overlay model predictions. Trial 1 is shown at the top and Trial 6 at the bottom.

Figs. 4, 5, and 6 show the predictions for the baseline, raw gaze, and overlay gaze models, respectively, along a time axis for each trial. In Figs. 4, 5, and 6, each row represents a different trial and each subtask is represented by a different color. The solid bottom line in each timeline is the ground truth and the prediction for each frame is represented above.

These plots provide qualitative insights regarding when in the trial and which subtasks the models performed poorly. Figs. 4, 5, and 6 show similarities and differences in the classification errors across the models. For example, in Trial 6 both gaze approaches commonly misclassified scoop food as dump food. On the other hand, the baseline approach

struggled to classify dump food correctly in the same trial. This example further emphasizes that, in order for gaze data to be useful for subtask classification, there must be sufficient quality control through careful gaze tracker calibration and confidence thresholding.

These figures also show where the addition of eye gaze improved the accuracy of subtask classification. For example, in Trial 3 the classification of the grasp spoon subtask improves for the raw gaze model, and improves even more for the overlay gaze model. We surmise from this result that the incorporation of eye gaze may produce the greatest improvement in accuracy in situations where the CGNL model on its own does not perform well, such as when a task is performed in an uncommon way. In real world applications, users are not confined to a set subtask order. To be useful in the context of shared-autonomy, the classification model must be robust to differences in user preferences and strategies.

V. Limitations and Future Work

A weakness of our study is the limited dataset used to evaluate our methods. This pilot study only had six trials from one able-bodied participant using only one kind of input interface. Within the context of teleoperation assistive robots, there is a large amount of variability in the strategies that individuals employ that our dataset did not address. These strategies vary based on many factors, such as user preferences, user experience, user's level of motor impairment, and the input interface (e.g. joystick, sip-n-puff, head array) [17]. For example, Javaremi et al found usage differences across groups using different input interfaces while completing the same teleoperated task, a result confirmed for both uninjured participants and participants with spinal cord injury. Future work should expand on our work by collecting and using a dataset that includes trials from multiple participants (uninjured and spinal cord injured) using a variety of input interfaces. We want subtask classification for shared autonomy to be robust to these real-world differences commonly seen in teleoperation.

Future work should also further investigate the effect of subtask order on the accuracy of each approach. It would be useful to compare the performance of each approach on several larger datasets that vary the ratio of trials with different subtask orders. This analysis would provide more insight into the effects of subtask order on classification accuracy and how eye gaze can potentially improve the classification of subtasks completed in a non-predominant order. We believe that continuing to investigate the accuracy gains produced for classification of non-predominant subtasks is important because we expect that users will have different preferences about subtask order. Thus, our models should be stress tested with datasets that have a variety of subtask orders.

Additionally, future work should explore other ways to featurize eye gaze. Our study was limited to two ways of incorporating eye gaze into the CGNL model. In future work, we plan to incorporate semantic gaze features into CGNL. Additionally, once we have collected a larger dataset, we want to implement and evaluate the 3D version of the CGNL model intended for action recognition along with other action recognition approaches. We chose to use CGNL as a starting point due to Saran et al's success incorporating raw gaze into it. However, we would like to explore other recent approaches in egocentric action recognition, especially ones that involve a gaze prediction step to better inform the final action classification. To our knowledge, none of these approaches have been evaluated on teleoperated tasks. We are interested to find a state-of-the-art approach that generalizes well or can made to generalize well to teleoperated activities, and we hope to investigate how different gaze features can make these approaches robust to the common differences in teleoperation that occur across users.

In conclusion, this work investigated using a promising action recognition approach with eye gaze data to classify subtasks in a multi-stage teleoperation task. We found that gaze data might make the approach robust to subtask orders that are underrepresented in the training data. These results have potentially significant real world applications because users of robot arms are unlikely to follow the same ordering of the subtasks required in a multi-task activity. Improving subtask classification, particularly to account for potential differences across users, is an important step towards scaling shared autonomy algorithms to multi-step activities. A shared control approach that can be used in multi-step tasks could make many activities of daily significantly easier to complete with an assistive robot and, in turn, increase the quality of life of its user.

ACKNOWLEDGMENT

This work was supported by the U.S. Air Force Academy, the Robotics Institute Summer Scholars and the National Defense and Science Engineering Fellowship. A special thank you to the coordinators of the RISS program for making this opportunity possible.

References

[1] K. Yue, M. Sun, Y. Yuan, F. Zhou, E. Ding, F. Xu, B. Vis, and B. Research, "Compact Generalized Non-local Network," Tech. Rep. [Online]. Available: https://github.com/KaiyuYue/cgnl-network.pytorch.

[2] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," in *International Journal of Robotics Research*, vol. 32, no. 7, 2013.

[3] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 6 2018.

[4] M. Land, N. Mennie, and J. Rusted, "The Roles of Vision and Eye Movements in the Control of Activities of Daily Living," *Perception*, vol. 28, no. 11, pp. 1311–1328, 11 1999. [Online]. Available: https://doi.org/10.1068/p2935

[5] T. Bader, M. Vogelgesang, and E. Klaus, "Multimodal integration of natural gaze behavior for intention recognition during object manipulation," in *ICMI-MLMI'09 - Proceedings of the International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interfaces*. New York, New York, USA: ACM Press, 2009, pp. 199–206. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1647314.1647350

[6] A. Saran, E. S. Short, A. Thomaz, and S. Niekum, "Understanding Teacher Gaze Patterns for Robot Learning," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 7 2020, pp. 1247–1258. [Online]. Available: http://proceedings.mlr.press/v100/saran20a.html

[7] Y. Li, M. Liu, and J. M. Rehg, "In the Eye of Beholder: Joint Learning of Gaze and Actions in First Person Video," in *ECCV: European Conference on Computer Vision*, 2018, pp. 619–635.

[8] Y. Huang, Z. Li, M. Cai, and Y. Sato, "Mutual Context Network for Jointly Estimating Egocentric Gaze and Actions," *IEEE Transactions on Image Processing*, vol. 29, pp. 7795–7806, 7 2020. [Online]. Available: http://arxiv.org/abs/1901.01874

[9] M. Lu, Z. N. Li, Y. Wang, and G. Pan, "Deep Attention Network for Egocentric Action Recognition," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 3703–3713, 8 2019.

[10] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, "Eye–Hand Coordination in Object Manipulation," *Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, 9 2001. [Online]. Available: https://www.jneurosci.org/content/21/17/6917

[11] M. Hayhoe and D. Ballard, "Eye movements in natural behavior," *Trends in Cognitive Sciences*, vol. 9, no. 4, pp. 188–194, 4 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364661305000598

[12] M. F. Land and M. Hayhoe, "In what ways do eye movements contribute to everyday activities?" in *Vision Research*, vol. 41, no. 25-26, 2001.

[13] R. M. Aronson, T. Santini, T. C. Kübler, E. Kasneci, S. Srinivasa, and H. Admoni, "Eye-Hand Behavior in Human-Robot Shared Manipulation," in *ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '18. New York, NY, USA: ACM, 2018, pp. 4–13. [Online]. Available: http://doi.acm.org/10.1145/3171221.3171287

[14] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local Neural Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[15] "Pupil Labs, Inc. 2017. Pupil Labs - Pupil. (2017)." [Online]. Available: https://pupil-labs.com/pupil/

[16] "http://www.kinovarobotics.com/assistive-robotics/products/robot-arms/." [Online]. Available: http://www.kinovarobotics.com/assistive-robotics/products/robot-arms/

[17] M. N. Javaremi, M. Young, and B. D. Argall, "Interface operation and implications for shared-control assistive robots," in *IEEE International Conference on Rehabilitation Robotics*, vol. 2019-June, 2019.

Trajectory Planning for Autonomous Vehicles Using Hierarchical Reinforcement Learning Under Noisy Observations

Kaleb Ben Naveed[1], Zhiqian Qiao[2] and John M. Dolan[2]

***Abstract*— Trajectory planning for autonomous vehicles is the process of computing the sequence of positions, velocities, and acceleration through space. Planning safe trajectories under uncertain conditions make the autonomous driving problem significantly complex. Current sampling-based methods such as Rapidly Exploring Random Trees (RRTs) are not well suited for the problem because of their high computational cost. Supervised learning methods such as Imitation Learning lack generalization and safety guarantees. To address these problems and in order to ensure a robust framework, we propose a Long-Short-Term-Memory (LSTM)-based Hierarchical Reinforcement Learning (HRL) structure for trajectory planning. Through HRL, the task of autonomous vehicle driving is divided into sub-goals and the network learns policies for both high-level options and low-level trajectory planner choices. The introduction of sub-goals decreases convergence time and enables the policies learned to be reused for other scenarios. In addition, the proposed planner is made robust by guaranteeing smooth trajectories and by considering the incomplete observations which result from the noisy perception system of the ego-car. Proportional–integral–derivative (PID) controller is used for tracking the points which helps to make the trajectory smooth and the problem of incomplete observations is dealt by using the LSTM layer in the network. We tested our framework in the high-fidelity CARLA simulator.**

***Index Terms*— Planning, Hierarchical Reinforcement Learning, Partially Observable MDPS, Lane Change.**

I. INTRODUCTION

Planning safe trajectories for autonomous vehicles is a challenging problem. In reality, this problem is particularly difficult because of the maneuver planning complexities and the incomplete observations coming from the noisy perception system of the car. While performing trajectory planning, autonomous car has to plan different maneuvers which might include following lane, waiting, changing lanes, and traversing intersections.

The existing methods for trajectory planning either relies on traditional classical planners or machine learning methods. Some of the state-of-the-art traditional planners include sampling-based planners, such as rapidly exploring random trees (RRTs) and lattice planners. RRTs randomly build a space-filling tree through space and are well suited for an environment with obstacles and differential constraints [1]. However, their computational complexity increases as the environment becomes more complex. Moreover, the trajectories generated are often not smooth for the ego-car to follow [2][3]. On the other hand, lattice planners are great at generating feasible paths and can incorporate constraints but graphs created might be incomplete, which leads to curvature discontinuity [2]. The other way of planning trajectories is through machine learning methods. The supervised learning method called Imitation learning [4] has shown some promising results. However, this method might not generalize well to the complex conditions and also there is no guarantee of stability and optimum solution [5].

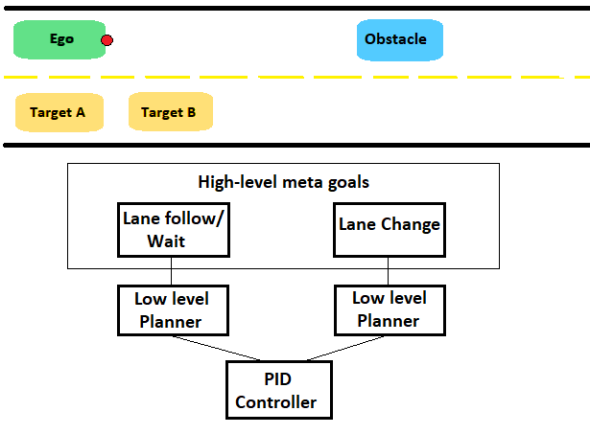


Fig. 1. The overview of the HRL structure and the scenario of lane change. The HRL structure has two two high-level options and a low-level trajectory planner for each high-level option. PID controller is used for following the planned trajectory

An alternative approach to the classical planners and data-robust methods is reinforcement learning (RL) [6]. The RL framework works on the principle of maximizing reward for a particular action at a given state. Existing RL work has shown promising results for an ego-car to learn policies for multiple scenarios. However, traditional RL methods for autonomous driving are less sample-efficient and less stable, especially for tasks with multiple sub-goals. In comparison to traditional RL, Hierarchical Reinforcement Learning (HRL) [7] allows model to learn policies for multiple sub-goals, which allows the policies learned to be reused for any other scenario. Furthermore, HRL has shown faster convergence rate which decreases training time for model to learn optimal policy. In this paper, we propose a Long-Short-Term-Memory (LSTM)-based HRL structure for trajectory planning. We choose the scenario of lane change shown in Figure 1 and give our structure two high-level options, which include Lane follow/Wait and Lane change. For each high-level planner, there is a separate low-level trajectory planner.

[1]Kaleb Ben Naveed is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong kaleb-ben.naveed@connect.polyu.hk

[2]Zhiqian Qiao and John M. Dolan are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, USA {zhiqianq, jdolan}@andrew.cmu.edu

The detailed description of low-level planner for each high-level option can be found in the methodology section. In order to make the proposed HRL trajectory planner more robust, we mainly worked on two problems.

- Planning safe and smooth trajectories: We used the proportional–integral–derivative (PID) controller for tracking trajectories instead of directly using low-level actions: throttle, steer, and brake
- Dealing with Noisy Observations: We used an LSTM layer in the network to help the model learn from sequences of observations, which compensated for missing observations and also helped to learn in more dynamic surroundings

II. RELATED WORK

A. Hierarchical Reinforcement Learning

By extending the framework of RL, [7] proposed the idea of hierarchical DQN in which the action value functions were integrated to operate at two levels of abstraction and learned a policy over meta goals and low-level actions. [8] proposed the concept of the Hierarchical Q-learning called MAXQ and showed that it produces better results than Q learning alone. In order to make the framework of hierarchical Q-learning more robust, [9] combined the R-MAX algorithm with MAXQ. This amalgamation added the safe exploration dimension of the model-based approach into the MAXQ. In the autonomous vehicle's domain, [10] used the three-layer HRL structure for options, low-level actions, and Q network for the decision making problem for traversing intersections. They showed faster convergence and improved results compared to non-hierarchical approaches. In order to further improve the sample efficiency and the reward structure, [11] proposed the state-attention model, the hybrid reward mechanism, and hierarchical prioritized experience replay. All these extensions improved sample efficiency and yielded higher results.

B. Work on Trajectory Planning and Prediction

Existing work on Trajectory Planning includes both classical traditional planners and planners using machine learning principles. One of the state-of-the-art sampling-based planner used in autonomous driving planning is RRTs [1]. RRTs generate trajectories by constructing a tree-like structure through the space. RRTs are proven good for environments with obstacles but might not converge to the optimal solution. [12] solved this problem by introducing RRT*, which showed optimal convergence and shorter routes. Another approach introduced was optimization-based lattice planners [2]. These planners were able to incorporate constraints and produce feasible paths. An alternative approach to classical planners includes supervised learning, which can be divided into imitation learning and trajectory prediction. [13] used a Deep Imitation learning framework to learn a driving policy for the urban scenarios through offline learning. They also added a safety controller module which increased the safety while testing.

For trajectory prediction, [14] proposed a system called UrbanFlow which consisted of a complete pipeline from collecting raw data to the final processing of trajectories. They used the UrbanFlow pipeline for trajectory prediction based on human drivers' driving behavior, which allowed the ego-car to make better decisions. Another approach that has produced significant results in trajectory prediction is Inverse Reinforcement Learning (IRL). IRL works on the principle of extracting the reward structure by observing an optimal trajectory of an expert agent. By building on IRL approach, [15] proposed a framework for predicting off-road vehicle trajectories by integrating kinematics and environment to recover the reward structure.

C. Partially Observable Markov Decision Processes (POMDP)

In order to deal with the incomplete observations from a noisy perception system, we model the problem as a POMDP. Some of the methods used for solving partially observable MDPs include finite state controllers, recurrent neural networks, and multi-agent reinforcement learning. Finite State controller [16] is a state-of-the-art rule-based method to solve POMDPs and has shown great ability to extract valuable information from the past noisy observation sequence. In the reinforcement learning domain, [17] extended the Deep Q-learning (DQN) model to Deep Recurrent Q-learning (DRQN) by replacing a fully connected layer with a LSTM layer. Moreover, they proposed bootstrapped random updates to train from a n-step sample from the replay memory. [18] further improved the DRQN model by adding the previous action to the tuple. This resulted in the Action DRQN (ADRQN) algorithm. By training from a previous action and observation pair, the results were improved for POMDPS. [19] used an alternative approach and solved POMDP using multi agents. They proposed the Multi Agent Deep Deterministic Policy Gradient method enhanced by a communication medium so that multiple agents can share their experiences and compensate for the missing information in other agents' observations.

D. Our Contribution

In this paper, we make the RL framework for trajectory planning more robust. The proposed HRL framework for trajectory planning gives the ego-car two high-level options and three low-level planner choices for each high-level option. We build our model on the works of [10][11] but instead of directly using throttle, steer, brake, we use a PID controller for tracking selected low-level trajectory choice. The HRL modular structure ensures reusability for the policies learned and has shown faster convergence rate. Trajectory tracking through PID resulted in smooth and stable maneuvers. In addition, in order to mitigate the problem of incomplete or inaccurate observations and more dynamic environments, we borrowed the idea of DRQN[17] and added LSTM layer in the hierarchical network structure.

III. PRELIMINARIES

A. Dueling and Double Deep Q-Learning

Double Q-learning is an extension of the state-of-the-art Deep Q Learning algorithm. The Q-learning algorithm in Reinforcement Learning is used to find an optimal action-selection policy π using a Q function which is used to maximize the action-value function $Q^*(s, a)$.

Deep Q-learning uses neural networks to update network parameter θ through minimizing the loss function between predicted action-value Q and the target action-value Y^Q. Deep Q learning uses the same values to select and evaluate an action, which results in overestimation.

Double Deep Q-learning solves the problem of overestimation by revising the target action updates from another target Q' network with different weights, which is shown in the Equation 1.

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a|\theta_t)|\theta_t^{'}) \quad (1)$$

The Dueling Networks breaks the evaluation of the Q function into two quantities: the value of being in state $V(s)$ and the advantage of taking action in state s, $A(s, a)$. After this the advantage function is normalized to address the issue of identifiability, as there is no unique way to identify V or A. This results in the Equation 2.

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{||a||} \sum_{a'} A(s, a^{'}) \quad (2)$$

B. Hierarchical Reinforcement Learning

HRL learns a policy at multiple levels as meta controller Q^1 generates the sub-goal g for the following steps and a controller Q^2 outputs the action a based on the sub-goal selected until the next sub-goal is generated by the meta-controller.

$$Y_t^{Q^1} = \sum_{t'=t+1}^{t+1+N} R_{t'} + \gamma \max_g (S_{t+1+N}, g|\theta_t^1) \quad (3)$$

$$Y_t^{Q^2} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a|\theta_t^2, g) \quad (4)$$

C. Partially Observable Markov Decision Processes (POMDP)

In the real world, the autonomous vehicles might have incomplete observations because of the noisy perception system of the car. This directly impacts the ability of system to plan trajectories. The POMDP is an extension of the MDP. An MDP is defined as a tuple $\{S, A, R, T, \gamma\}$. S is the state space, A is the available actions, R defines the immediate reward for each state-action pair. T is the transition probability that maps an action pair (s, a) to a new state and gamma is the discount factor between $[0, 1]$. POMDP is defined as tuple $\{S, A, R, T, \pi, V, \gamma\}$; it adds the set of environment observations π and a set of conditional observations V.

IV. METHODOLOGY

A. Scenario and Problem Description

In this paper, we chose the scenario of lane change with added complexity. The high-level overview of the scenario can be seen in Figure 1. The ego-car (green car) starts in the ego-lane (lane that has the ego-car) and is required to do a lane change, as there is an obstacle (blue car) in front of it. But the target-lane (the lane where the ego-car chooses to go) has an inflow of traffic in it. In our case, we are using two cars, colored in yellow to represent the surrounding traffic. This scenario can be easily broken down into tasks with multiple sub-goals which explains our reason for choosing this scenario. In this problem we are providing our car with two high-level options for the maneuver. The first option is lane follow/wait and the second option for the ego-car is the lane change maneuver. The high-level structure of the proposed HRL trajectory planner can be seen in Figure 1.

B. HRL-based Trajectory Planner

Through HRL we divide the task of lane change into two high-level options: lane follow/wait and lane change. This helps us to plan trajectories for each option separately. Through HRL, the ego-car learns a policy for both high-level options and for the low-level trajectory planners. The network used is Deep Dueling Double Q Network for both the high level options network and the low level trajectory planner network. The LSTM-layer is used alongside fully-connected layer in both the networks. The detailed working of the high-level options network and the low-level planner network is mentioned below:

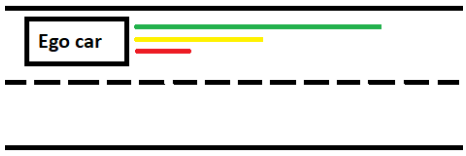


Fig. 2. The overview of lane follow/wait option

1) Lane Follow/Wait: Once the ego-car selects the lane follow/wait option, the low-level trajectory planner is used to plan the path. The low-level planner has three choices for the trajectory, as shown in the form of coloured lines in Figure 2. The green line represents the choice in which the ego-car can lane-follow for greater distance. This choice is rewarded when the ego-car does not observe any obstacle within certain radius of its current position in the ego-lane. The yellow line represents a trajectory choice under which the ego-car can lane-follow for a shorter distance. This choice is beneficial when ego-car observes an obstacle car in the ego-lane at some distance and avoids getting close to it. On the other hand, if the red line trajectory is chosen by the ego-car, it will decelerate to the wait option. Once the trajectory selection is made, the PID controller will be used to complete the path planned. In order to reduce jerk resulting from poor selection of low-level trajectory choices, we tailored the reward structure mechanism, which can be found in the subsequent section.

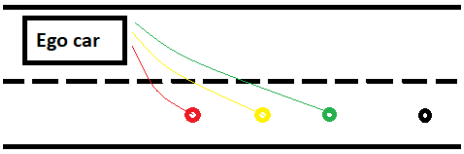


Fig. 3. The overview of lane change option

2) Lane Change: For the lane change option, the ego-car is given three points as low-level choices for the trajectory. The overview of this option can be seen in Figure 3. Each different point represents a different velocity profile the ego-car may choose to make a lane change. This ensures safety and stability while planning for the lane change maneuver. The green point is selected if the ego-car is required to do a faster lane change because of the higher speed in the ego-lane. The yellow point is for the normal velocity profile trajectory. The red point selection helps the ego-car to take sharper turns, which might be needed when the velocity of the ego-car is significantly less or it was in the wait option. The RL selects one point between the three points through low-level planner network. Once the policy selects the point, a black point is generated at some distance in the target lane. This black point is a safety follow point which helps the ego-car to reorient itself before the lane follow/wait option is activated again.

C. Dealing with Noisy Observation

In order for our model to consider incomplete observations and random scenarios, we added a LSTM layer for both options and the low-level planner network. We used three time steps as an input to the LSTM layer. In order to sample experiences from replay memory, we used the approach of bootstrapped random updates proposed by [17]. This strategy randomly samples an n step sequence from the batch of episodes drawn from experience replay and then trains the neural network. In order to test our model with incomplete observations, we added noise to the state space, which includes the information of other cars available to the ego-car.

D. State Space

In order to formulate the state space for our Hierarchical Structure, we used information from the ego-car (e), obstacle car (o), target car A (a), and target car B (b) in the given scenario. This can be seen in the tuple given below. The safe range for the ego-car is $x >= 16$ for the stopped vehicle and $x >= 7$ for the moving vehicles, where x is the ego-car distance to other vehicles. The state space consists of 18 parameters. The tuple contains the state information where $f \in \{o, a, b\}$.

$$s = [v_e, lane_{ide}, v_f, d_{ef}, d_{cf}, d_{cxr}, lane_{idf}, d_{eg}]$$

- v_e = Ego-car velocity
- $lane_{ide}$ = Lane ID for ego-car
- v_f = Velocities of other vehicles
- d_{ef} = Ego-car distance to other vehicles
- d_{cf} = Ego-car chase distance to other vehicles
- d_{cxr} = Ratio of chase distance to safety threshold
- $lane_{idf}$ = Lane ID for other vehicles
- d_{eg} = Ego-car distance to goal position

E. Reward Structure

We build our reward structure on the hybrid reward structure proposed by [11]. Under the hybrid reward structure, a separate reward is given for high-level option selection and low-level trajectory choice. The ego-car gets penalised for choosing wrong option and choosing wrong low-level trajectory choice separately. The low-level choice gets penalised if the chosen trajectory leads to unsuccessful completion of the sub-goal, which in our case is collision with one of the target cars or the obstacle car. Moreover, in order to plan safer and smoother trajectory, low-level choices were penalised if they were not required. For example, under option 1, low-level wait choice gets penalised if it is selected unnecessarily. Also ego-car is given positive reward for maintaining safe distance to the ego-car. The safe distance is the minimum distance, which ego-car has to maintain from the obstacle car. Ego car is expected to plan a longer trajectory if it does not see any obstacle in the ego lane within certain radius. For lane change maneuver, ego-car gets a reward for the safest point selected based on its past velocity profile. For example, if the ego-car has a higher velocity in the target-lane, it is given higher reward to do a faster lane change; otherwise if ego-car was waiting in the ego lane, it is expected to do a slower turn. These different low-level point choices prevent lane-invasion and unnecessary jerk.

V. Experiments

In this section, we apply the proposed LSTM-based HRL algorithm to the scenario shown in Figure 1. in the CARLA simulator with 30 fps. The scenario is the lane change scenario with added complexity. The ego-car is expected to perform either the lane-follow or lane-change maneuver when required. The ego-car has to avoid collision with the obstacle in the ego-lane and the traffic in the target-lane. In order to test the generality and safety of our approach, we made surroundings of the ego-car more dynamic. Following two points describe details about how we made the environment dynamic.

- Randomly changing the target cars' velocity
- Randomly changing the position of the Obstacle car over a certain distance

In addition, in order to evaluate the performance of our algorithm with inaccurate or incomplete observations, we added noise to the state space. In particular, the noise was added to the ego-car distance to the target cars and the obstacle car. Mentioned below are the two proposed algorithms responsible for selecting options and for planning trajectory under each high-level option. Algorithm 1 describes the proposed LSTM-based Hierarchical Structure. Algorithm 2 describes the low-level trajectory planner, which is responsible for selecting low-level trajectory choice given high-level option.

Algorithm 1 LSTM-based HRL

Initialize options and planner network Q^o, Q^p with weights θ^o, θ^p
Initialize target options and target planner network $Q^{o'}$, $Q^{p'}$ with weights $\theta^{o'}$, $\theta^{P'}$.
Construct empty replay buffer B with max memory length l_b
for 0 to E training episodes **do**
  Get Initial States S.
  Reset hidden state for options network $h^O \leftarrow 0$.
  Reset hidden state for planner network $h^P \leftarrow 0$.
  while s is not terminal state **do**
    O_t and h^O = $\arg\max_o Q(h^O_{t-1}, S_t)$ based on the $\epsilon - greedy$, where O_t is the option.
    P_t and h^P = $\arg\max_p Q(h^P_{t-1}, S_t, O_t)$ based on the $\epsilon - greedy$, where P_t is planner choice.
    Pass option and planner choice to $TrajectoryPlanner(O_t, P_t)$.
    Get new state S_{t+1}.
    Get R^o_{t+1}, R^p_{t+1}, where R^o is the option reward and R^p is the planner reward.
    Store transition T into B: $T = \{S_t, O_t, P_t, R^o_{t+1}, R^p_{t+1}, S_{t+1}\}$.
  end while
  Train with Buffer $RelayBuffer(e)$.
end for

Algorithm 2 TrajectoryPlanner

if $Option == LaneFollow/Wait$ **then**
  if $planner == 1$ **then**
    Wait
  else if $planner == 2$ **then**
    follow lane for m distance
  else if $planner == 3$ **then**
    follow lane for n distance, where n $<$ m
  end if
else if $Option == LaneChange$ **then**
  if $planner == 1$ **then**
    Slower lane change
  else if $planner == 2$ **then**
    Normal speed lane change
  else if $planner == 3$ **then**
    Fast Lane Change
  end if
end if

VI. Results

The advantages of the proposed HRL-based planner include safety and reusability of low-level planner under a given option. The safety in our approach was achieved through the HRL modular structure and the use of the PID controller. The demonstration of the ego-car trajectory is shown in figure 4. The policy learned is from the incomplete observations which resulted from missing information of other cars in the state tuple.

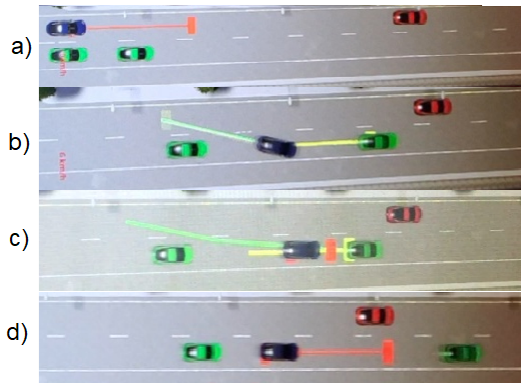


Fig. 4. It is the sequence of four image frames representing different time steps (from top to bottom) of the ego-car's trajectory in the CARLA simulator. In the first frame a, the ego-car is performing lane-follow. In the next couple of frames, b and c, the ego-car is performing lane change maneuver. In frame d, the ego-car again starts to follow the lane.

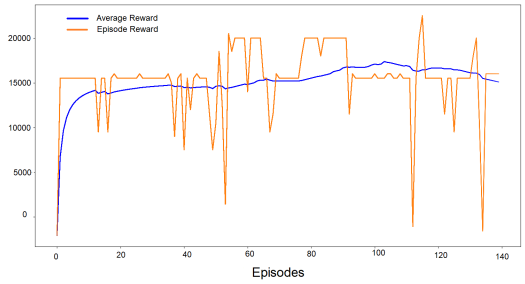


Fig. 5. Reward Graph for the test run for 140 episodes. The blue line shows the average reward and orange line shows the episode reward.

The red line trajectory shows that the car is in the lane follow/wait option while the yellow color trajectory shows the lane change maneuver. The red line at the start of the trajectory shows that car choose to follow lane for longer distance as the obstacle car is far from the ego-car current position. As soon as ego-car observed that the distance to the obstacle is decreasing and the target lane is also free for a lane change, it opted for the lane change option. For the lane change, the yellow-coloured line trajectory is generated, which was followed by the ego-car. Another thing to consider in this scenario is that once the ego-car does a lane change it ends in between two target cars. In this case, ego-car is expected to adjust its speed in order to safely complete a lane change and then to follow the lane. It is shown in Figure 4 that right after changing lane, the ego-car started to decelerate in order for the front target car to move some distance. Then after this, ego-car started to plan longer trajectories as distance between front and ego-car increased. The deceleration after lane change ensured safety and prevented the ego-car from crashing into the front vehicle.

The preliminary results suggest that the proposed approach guarantees safety by selecting different low level trajectories choices. The PID controller helped car to have stable and trajectories and smooth maneuvers as the lane-invasion readings were almost zero with final policy after 1600 episodes.

After training for 1600 episodes, the policies learned for both high-level network and low-level planner network were tested. The test run included 140 episodes. Figure 5 shows the graph of the episode and average reward. The overall success rate during the test run was almost 95 percent.

VII. FUTURE WORK

The future work for the proposed LSTM-based HRL planner include extensive testing with more dynamic surroundings and more noise in the state space. Preliminary results suggest that the proposed planner can handle dynamic surroundings but the collision rate is still high. Some of the areas for improvement include reward structure and network structure. In addition, we also hope to evaluate the possibility of combining the already learned policies with the new scenarios, which might include traversing intersections and lane merging. Moreover, we hope to do extensive testing in the partially observable environments so that our approach can be verified in more dynamic and real world settings.

ACKNOWLEDGMENT

I (Kaleb) would like to express my gratitude to the management of the Robotics Institute Summer Scholars (RISS) Program for providing me with this valuable research experience. I would also like to thank RISS program leadership especially Ms Rachel Burcin for her efforts in making RISS possible this summer. I am also grateful to my mentors, Dr John Dolan and Ms Zhiqian Qiao, for their continued support throughout the RISS program. Likewise, I would like to thank the Industrial Center, The Hong Kong Polytechnic University, for providing me with resources for remote research.

REFERENCES

[1] S. LaValle, J. Kuffner, and B. Donaldetal, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and computational robotics: new directions*, 2001, pp. 293–308.

[2] H. Mouhagir, R. Talj, V. Cherfaoui, F. Guillemard, and F. Aioun, "A markov decision process-based approach for trajectory planning with clothoid tentacles," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 1254–1259.

[3] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2061–2067.

[4] P. Cai, Y. Sun, Y. Chen, and M. Liu, "Vision-based trajectory planning via imitation learning for autonomous vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2736–2742.

[5] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, Mar. 2018.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: MIT Press, 1998.

[7] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 3675–3683.

[8] T. G. Dietterich, "The maxq method for hierarchical reinforcement learning." *ICML*, vol. 98, pp. 1–179, 1998.

[9] N. K. Jong and P. Stone, "Hierarchical model-based reinforcement learning: R-max+ maxq," *in Proceedings of the 25th international conference on Machine learning. ACM*, pp. 1–179, 2008.

[10] Z. Qiao, K. Muelling, J. Dolan, P. Palanisamy, and P. Mudalige, "Pomdp and hierarchical options mdp with continuous actions for autonomous driving at intersections," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2377–2382.

[11] Z. Qiao, Z. Tyree, P. Mudalige, J. Schneider, and J. Dolan, "Hierarchical reinforcement learning method for autonomous vehicle behavior planning," Nov. 2019.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] J. Chen, B. Yuan, and M. Tomizuka, "Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety," 2019.

[14] Z. Qiao, J. Zhao, Z. Tyree, P. Mudalige, J. Schneider, and J. Dolan, "Human driver behavior prediction based on urbanflow," 2019.

[15] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, "Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories," *CoRR*, vol. abs/1810.07225, 2018.

[16] N. Meuleau, L. Peshkin, K. E. Kim, and L. P. Kaelbling, "Learning finite state controllers for partially observable environments." *15th international conference of uncertainty in AI*, pp. 427–436, 1999.

[17] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," 2015.

[18] P. Zhu, X. Li, P. Poupart, and G. Miao, "On improving deep reinforcement learning for pomdps," 2017.

[19] O. Kilinc and G. Montana, "Multi-agent deep reinforcement learning with extremely noisy observations," 2018.

Development and Analysis of a Multi-Agent Incomplete Information Task with the Multi-Agent Deep Deterministic Policy Gradient Algorithm

Beverley-Claire Okogwu[1], Mengdi Xu[2], and Ding Zhao[3]

***Abstract*— Deep Reinforcement Learning (DRL)[1] has become a popular Machine Learning property in recent years due to the establishment of Deep Neural Networks [2]. DRL has also been introduced into numerous applications which span across various fields such as healthcare, industry, computer vision, natural language processing, and machine translation [3]. However, as it is a rather new field, DRL is prone to certain biases and unprecedented situations. One of these uncertain properties is ambiguity in a multi agent system, such as with Human-Robot Interaction (HRI) [4]. In this paper, we look at how DRL algorithms are applied in HRI using Deep Learning techniques such as the Actor-Critic algorithm and the Multi Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. We will then utilize this algorithm to train agents in our newly proposed multi agent environment that supports this model in HRI. Furthermore, we consider training via an ensemble of Neural Networks to examine uncertainty and establish consistency.**

***Index Terms*— Deep Reinforcement Learning, Human-Robot Interaction, MADDPG, Soft Actor-Critic, Multi-Agent Systems.**

I. INTRODUCTION

In recent years, Deep Reinforcement Learning (DRL) has played a significant role in Human-Robot Interaction, particularly with the ability for a robot to emulate certain social skills such as communication, navigation, cooperation, and coordination [1]. In developing such skills, the behavior of the robot must be controlled via a system of rewards/punishments, which follow a variety of DRL frameworks and predictive models like the intrinsically motivated rewards system [5].

There exist a variety of DRL algorithms, some of which include Deep Q Neural Networks, Policy Gradient, and Q-learning algorithms [2]. These algorithms each have different effects on the system, which depend on changes to factors such as the definition of the agents' reward, state, action, and policy.

However, since DRL is a relatively new field, it is prone to certain biases and unforeseen situations. The novelty of these DRL algorithms leads to natural limitations and biases, hindering their ability to predict ambiguity. This particularly impacts multi agent environments[6] with respect to cooperative, competitive, and mixed environments [7]. Addressing this issue is important to ensure the system is void of errors and works at the optimal level.

There exist numerous literature that have looked into ways to solve such problems, and the Actor Critic algorithms have emerged as a first step in overcoming the errors. The Soft Actor-Critic (SAC) algorithm [8], mainly used in Human-Robot Collaboration via Deep Learning [9], stands out among RL algorithms due to its automatic temperature tuning. With this tuning, the number of trials in a testing period are significantly reduced, thus minimizing the total time taken to completely train an Artificially Intelligent agent to perform a certain task.

Although SAC is a popular algorithm in a single agent system, it is still prone to errors in a complex multi-agent system [10], especially where some degree of communication is required for the system to work. Lowe et al [11] overcome this by developing a new kind of algorithm, the Multi Agent Deep Deterministic Policy Gradient (MADDPG) algorithm, adapted from the SAC. MADDPG differs from the Deep Deterministic Policy Gradient (DDPG) algorithm with respect to the type of critic used: MADDPG uses a centralized critic with a decentralized action, as opposed to DDPG. With this setup in MADDPG, the critic has access to more than enough information regarding the policies of additional agents in the multi-agent system.

In this paper, we adapt the MADDPG algorithm and build a new environment. What makes our environment different from [11] lies in the definition of the agents. In [11], two main types of agents are defined: the regular agents, which cooperate, and the adversaries, which try to hinder or compete with the agents. In our environment, we define two types of sub-agents that either cooperate or compete with the adversary itself: the ally and the enemy. We will be using these "sub-agents" to experiment and test with the MADDPG to ensure consistent results. In addition to the vanilla MADDPG, we will thus propose a way to train our agents via an ensemble of Neural Networks(NN)[12] to further analyze uncertainty.

SUMMARY OF CONTRIBUTIONS

- II. We address important background information relevant to paper.

[1]Beverley-Claire Okogwu is with the Department of Computer Science and Mathematics, Dickinson College, Carlisle, Pennsylvania okogwub@dickinson.edu, beverley.okogwu@gmail.com

[2]Mengdi Xu is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania mengdixu@andrew.cmu.edu

[3]Ding Zhao is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania dingzhao@andrew.cmu.edu

- III. We outline and describe the steps taken in constructing the environment.
- IV. We analyze the results from training our agents via the MADDPG algorithm.
- V-VI. We propose ensemble NN in training.

II. BACKGROUND

In order to fully comprehend the techniques involved behind the environment, we must first address certain Reinforcement Learning concepts:

A. Reinforcement Learning, RL

We can think of Reinforcement Learning as a reward/penalty system as shown in Figure 1 below:

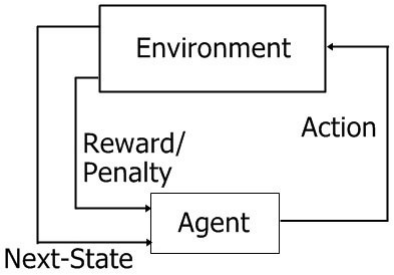


Fig 1: General Reward/Punishment scheme for Reinforcement Learning [13].

Here, each agent performs a given action in an environment. The agent, in turn, is rewarded or penalized based on the action performed. This is repeated at every state until the return/goal is maximized. We can thus think of RL as a way in which the agent continuously explores and exploits the environment to obtain the highest return. This can be formalized as a finite Markov Decision Process (MDP), where the sets of states, actions, and rewards all have finite number of elements.

Appendix *A* addresses the formalization of a finite MDP [13].

B. Ambiguity

To elucidate the concept of ambiguity, we must first describe how an ambiguous situation is met. If the robot is given a set of instructions/information, it will carry out its tasks ad hoc. However, there are instances where the robot is given incomplete or no information at all. In this case, the robot will rely on ambiguous uncertainty [14]. Analogously, the same behavior is observed in humans: if a person is travelling on a known route and unexpectedly arrives at an unknown fork, the individual would have to make a judgement on which path to take, which may be open to multiple interpretations. Note that there are differences in how each individual would handle these ambiguous situations, hence the ambiguity sensitivity [15].

In this paper, ambiguity will refer to the adversary agent's uncertainty about the type of the other agents. This will be discussed in detail in the methods section of this paper.

C. MADDPG

Originally proposed by Lowe et al [11], the MADDPG algorithm as shown in Figure 2, makes use of a centralized critic with the addition of a decentralized execution. This simply means that the policies can use additional information relating to other agents in the system to make the training much easier. This is done while still independently taking actions and recording observations.

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)\big|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i(o_i^j)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  end for
end for

Fig 2: The MADDPG Algorithm [11].

[8] then experiment with their algorithm on a variety of Cooperative, Competitive, and mixed environments as shown in Figure 3, which run against other RL algorithms as well. Their results show the superiority of the MADDPG algorithm over the DDPG and similar algorithms.

Fig 3: Results from running MADDPG Algorithm [11].

D. Neural Networks

Following the training of agents in our newly built environment, we would also train via an ensemble of Neural Networks (NN). This simply means that we will run multiple instances of the MADDPG algorithm on the environment and take an average of the results. This process is shown in Figure 4:

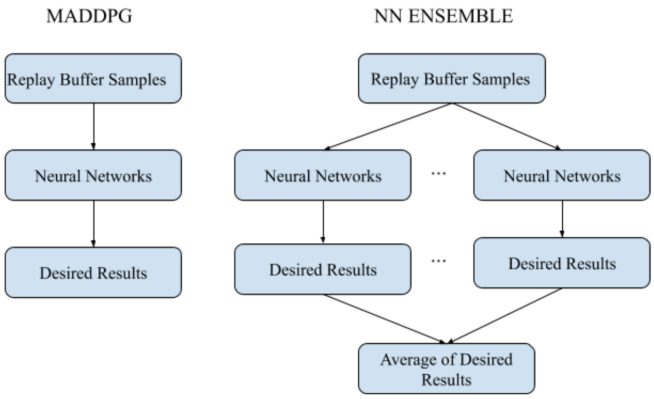


Fig 4: Big Picture behind Neural Network Ensembles.

III. METHODS

In developing our environment, a few factors were considered:

- The platform to be used
- The behavior of the agents
- The reward of the agents

A. Platform

For our environment, we used AI Gym, a free open source application available online. In addition to Gym, we utilized a pyTorch implementation of the maddpg algorithm [16] in training and experimenting with the environment. All codes were done in Python. Additional requirements included TensorFlow and OpenAI baselines.

B. Agent Behaviors

After determining suitable platforms, we then proposed ways by which the agents would interact with each other, as well as the general outlook of the environment:

- The environment takes two agents at a time.
- One agent is always an adversary (colored blue).
- The other agent can either be an enemy or an ally to the adversary.
- There is a 50 percent chance of the agent being an enemy and a 50 percent chance of the agent being an ally.
- If the agent is an ally, the agent will be colored green, and its goal is the green landmark.
- If the agent is an enemy, the agent will be colored red, and its goal is the red landmark.

Recall, from our definition of ambiguity, that the adversary remains unaware of the type of sub agent (ally or enemy). This is where the adversary has to make guesses and infer information based on the behavior of the sub agents.

C. Rewards

In each episode of the environment, the agent made an attempt to reach its target goal (red or green as described in part *B*). Thus, the agents were rewarded according to their performance in attaining the desired goal. However, there were additional factors to consider while rewarding agents:

If the agent was an ally, the agent did not need to avoid the adversary; it only needed to reach the green goal. Thus, the agent and adversary were both rewarded by minimizing the distance between themselves and the landmark. Figures 5a and b show snapshots of the environment being run where the sub agent is an ally and an enemy respectively.

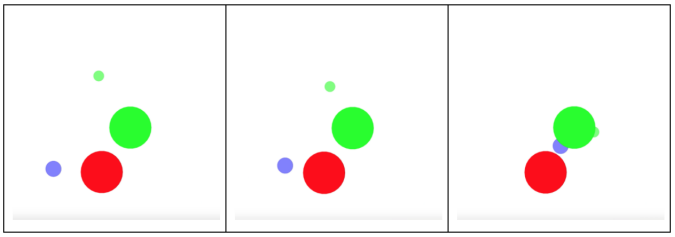


Fig 5a: Environment with an adversary and its ally

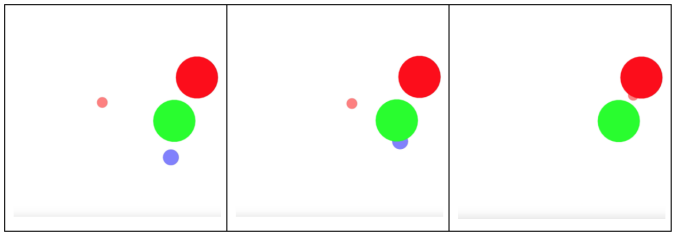


Fig 5b: Environment with an adversary and its enemy

If the agent was an enemy, then the reward system differed slightly. The adversary was positively rewarded by minimizing the distance between itself and the enemy. The enemy was positively rewarded by minimizing the distance between itself and the red landmark while maximizing the distance between itself and the adversary. In this case, the system became similar to the predator-prey model discussed in [11].

Note that the enemy's reward system prioritizes avoiding the adversary over reaching the red landmark similar to a rabbit evading a fox before going home.

IV. RESULTS

Results showed the environment worked well with the agent pairs (adversary, ally) and (adversary, enemy). All codes and videos of the environments being run can be seen on our google sites: *https://sites.google.com/view/riss2020multiagentenv/home*

While the environment was running, we recorded the average reward of each agent plotted against the number of episodes as shown in Figure 6. For training purposes, the environment was run for 25,000 episodes.

Note that agent0 (purple) is the adversary and agent1(orange) is the sub agent.

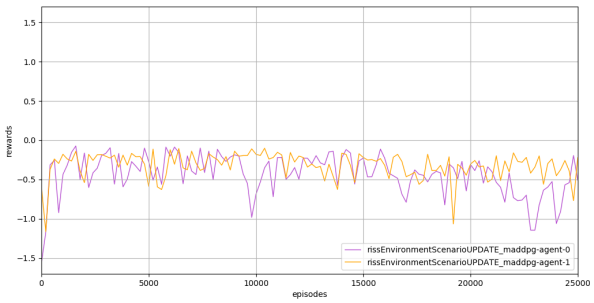


Fig 6: Agents' Reward vs Total number of episodes run

From the graph, we can infer the following:

- An increasing slope in the first few episodes show both agents' improvement while learning.
- The sub agent learns better than the adversary. This is because the adversary has to make more guesses about the sub agent's type, which leads to a slower and more hindered learning process.
- For both agents 0 and 1, there is an obvious fluctuation gap observed between 20,000 and 25,000 episodes. An investigation into the cause of such fluctuation is outside the scope of this paper.

V. CONCLUSION AND FUTURE WORK

We have seen how Deep Reinforcement Learning algorithms like MADDPG help improve the performance of multi-agent systems, and we have built and ran our environment on this algorithm to further analyze if sub agents can adapt as well. We have also further analyzed our environment to understand how the adversary agent deals with ambiguity in the environment. We have thus shown that using purely MADDPG is not sufficient to handle the ambiguity about the agent types in incomplete information games. We will leave the development of MADDPG-based algorithm that can safely handle the ambiguity for future work.

In addition, we hope to further analyze more of such sub agents that when applied to our NN ensemble, would get an uncertainty score very close to 0.

APPENDIX

A. Formalization of a Finite Markov Decision Process

Let the set of states, actions, and rewards be expressed as thus (S, A, R). We can further state that the set contains a finite number of elements.

Therefore, for certain values of some random values R_t and S_t, namely some $s' \in S_t$ and $r \in R_t$, there exist a probability for those values to occur at a time, t with respect to a certain preceding state, s and action, a:

$$p(s', r|s, a) = Pr(S_t = s', R_t = r|S_t - 1 = s, A_t - 1 = a)$$

Note that in an MDP, the probability of each possible value for R_t and S_t solely depends on *only* the preceding state and action.

$$p(s'|s, a) = Pr(S_t = s'|S_t - 1 = s, A_t - 1 = a) = \sum_{r \in R} p(s', r|s, a)$$

Similarly, the expected returns/rewards for state-action pairs can be computed as a two argument function

$r : S \times A \times S \to R$

$$r(s, a) = E(R_t|S_t - 1 = s, A_t - 1 = a) = \sum_{r \in R} r \sum_{r \in R} p(s', r|s, a)$$

ACKNOWLEDGMENTS

This work would not have been possible without the support of the Robotics Institute Summer Scholars Program and the sponsorship of Mobility 21 scholarship. Special thanks to Dr. Ding Zhao and Mengdi Xu from Carnegie Mellon University's Safe AI Lab for their mentorship and support, especially Mengdi Xu for providing the reward plot code and the overall research outline. We would also like to thank Rachel Burcin and Dr.John Dolan for tirelessly supporting us throughout the summer.

REFERENCES

[1] K. Arulkumaran, M. Deisenroth, M. Brundage, A.Bharath. A Brief Survey of Deep Reinforcement Learning. *IEEE Signal Processing Magazine*, vol.34, pages 26-38, Nov, 2017.
[2] J. Schmidhuber, Deep Learning in Neural Networks: An Overview. *Science Direct*, vol. 61, pages 85 – 117, 2015.
[3] Y. Li, Deep Reinforcement Learning: An Overview. *Machine Learning*. 2018.
[4] Figure 1. Reinforcement Learning Block Diagram. Illustration from The Research Gate.
[5] M. Goodrich, A. Schultz. Human-robot interaction: a survey. *Now Publishers Inc*, 2008.
[6] A. Qureshi, Y. Nakamura, Y. Yoshikawa, H. Ishiguro. Intrinsically Motivated Reinforcement Learning for Human-Robot Interaction in the Real-World. *Osaka University*. 2018.
[7] S.Aaron, et al. "Common metrics for human-robot interaction." *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. 2006.
[8] D.W.Johnson, R. T. Johnson. "Cooperative, competitive, and individualistic learning environments." *International guide to student achievement*, pages 372-374, 2013.
[9] T. Haarnoja, A. Zhou, et al. Soft Actor-Critic Algorithms and Applications. Machine Learning, 2019.
[10] J. Tjomsland, A. Shafti, A. Faisal. Human-Robot Collaboration via Deep Reinforcement Learning of Real-World Interactions. UKRI CDT for AI in Healthcare, Imperial College London 2019.
[11] K. P. Sycara, "Multiagent Systems ", *AIMag*, vol. 19, no. 2, p. 79, Jun. 1998.
[12] L.Ryan, Y. Wu, A. Tamar, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *In Advances in neural information processing systems*, pp. 6379-6390. 2017.
[13] N. Ury, N. Intrator, and D. Horn. "Optimal ensemble averaging of neural networks." *Network: Computation in Neural Systems*, vol 8, no. 3, pages 283-296, 1997.
[14] R.S. Sutton, A.G. Barto. "Reinforcement Learning: An Introduction", 2nd edition, *The MIT Press*, 2018.
[15] R. Cubitt, G. van de Kuilen, and S. Mukerji. The strength of sensitivity to ambiguity. *Theory and decision*, 85(3-4):275–302, 2018.
[16] A. Jungmann, F. Mohr and B. Kleinjohann, "Applying Reinforcement Learning for Resolving Ambiguity in Service Composition," in 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications (SOCA), Matsue, Japan, 2014 pages. 105-112.
[17] S. Iqbal, 2018, "MADDPG-Pytorch", Shariqiqbal2810. https://github.com/shariqiqbal2810/maddpg-pytorch

Open-World 3D Detection and Tracking in Autonomous Driving

Neehar Peri[1], Achal Dave[2], Deva Ramanan[2,3] and Shu Kong[2]

***Abstract*— Safety-critical perception systems, such as autonomous vehicles, must be aware of a diverse number of objects for safe operation in the open-world. Multi-object detection and tracking (MOT) is a key component in such systems, and is typically instantiated by training on a closed set of known categories (e.g., pedestrians and vehicles). As a result, the trained MOT model cannot detect and track *unknown* objects belonging to other classes not seen during training. For operational safety, detecting and tracking these salient unknown objects is undoubtedly crucial. Therefore, we study open-world MOT in that we train a MOT model to detect and track objects not only from *known* classes (as is typically done in literature) but also unknown objects belonging to other open-world classes. Specifically, we focus on 3D MOT using LiDAR point clouds that provide direct and accurate 3D measurements. We extend existing 3D MOT evaluation protocols to this open-world setting by additionally evaluating on an open-world "unknown" super-class. We re-purpose annotated uncommon classes (that are typically not used in either training or testing) in the nuScenes dataset to benchmark performance on open-world "unknown" objects. We demonstrate that our approach significantly improves MOT robustness on unknown objects, while also improving state-of-the-art performance on known classes. We also qualitatively show that our proposed method is able to detect real unknown objects (e.g., temporary signs).**

***Index Terms*— Open-World Perception, LiDAR, 3D Detection, Multi-Object Tracking, Autonomous Vehicles.**

I. Introduction

Safety-critical systems such as autonomous vehicles must reliably understand the surrounding environment for safe and accurate operation in the *open-world* [1], [2]. Multi-object detection and tracking (MOT) is a critical first step for many perception pipelines, enabling higher-level modules such as motion forecasting and planning [3].

MOT has seen significant progress in recent years [4]–[7], driven by state-of-the-art (SOTA) approaches based on neural networks that learn directly from sensor data (e.g., LiDAR and radar for 3D MOT in autonomous driving) [3], [6], [7]. However, such models train on and operate in a "closed-world" that consists of only a selected number of common classes [4], [5], which are believed to be safety-critical (e.g., pedestrians and vehicles). Inevitably, MOT models that are trained using a limited "closed-world" ontology cannot detect and track objects from previously unseen (i.e *unknown*) objects [2]. Identifying and tracking these unknown instances, as shown in Fig. 1, can be critical to safe operation.

[1] Neehar Peri is with the Center for Automation Research, University of Maryland peri@umiacs.umd.edu

[2] Achal Dave, Deva Ramanan, and Shu Kong are with the Robotics Institute, Carnegie Mellon University {achald, deva, shukong}@andrew.cmu.edu

[3] Deva Ramanan is with ArgoAI

Motivation. To address such unknown objects, we study MOT in the *open-world*: we train models with both closed-world, known classes (as typically done in literature) and a limited number of unknown objects to define a catchall class and evaluate performance on both *known* classes and unseen *unknown* objects. Admittedly, enumerating *unknown* taxonomic classes for evaluation is prohibitively challenging, as there may not be enough examples due to the long-tail distribution encountered in natural data [8]. Therefore, we can instead evaluate over an "unknown" *super-class*, which captures other meaningful objects not included in traditional closed-world MOT and are thought to be critical. To explore this novel open-world setup, we establish formal evaluation protocols, introduce a benchmark setup over an existing well-curated dataset (nuScenes [6]), and present a simple yet effective method that not only improves upon SOTA MOT performance on closed-world classes, but also faciltates the ability to detect and track unknown objects.

Contribution 1: *We explore open-world 3D MOT under a realistic and practically meaningful scenario, considering both known and unknown objects in the context of autonomous driving*. Typically, open-world problems are studied using synthetic setups, such as image classification on MNIST with unknown testing examples [9], [10]. Within such synthetic setups, it is hard to draw meaningful conclusions applicable to real-world scenarios. In contrast, we place ourselves in the open 3D world, and explore open-world 3D MOT (based on LiDAR input), which embraces 3D detection and tracking. Studying open-world MOT cannot be trivially extended from typical (closed-world) MOT due to some inherent challenges, as it requires careful consideration of the definition and evaluation of open-world unknown objects.

Contribution 2: *We introduce meaningful evaluation metrics and benchmarking protocols for open-world MOT*. By design, MOT methods search spatially over the input data (stacked LiDAR sweeps in our case), and localize objects of interest according to some predefined closed-world classes. Therefore, the first challenge in open-world MOT is to define the set of *open-world unknown objects*. While it is straightforward to collect examples from closed-world classes, it is not trivial to curate open-world *unknown* objects, which should not be from either closed-world classes or the background class, but rather should be from other classes of interest (e.g., animals, movable objects and debris on the road). Interestingly, we realize that modern datasets such as nuScenes [6] already annotate such classes (e.g., animals, debris and other movable objects) but never exploit them for training and testing, presumably because there are not enough examples to train class specific detectors. (cf. Fig. 3). In order to effectively study open-

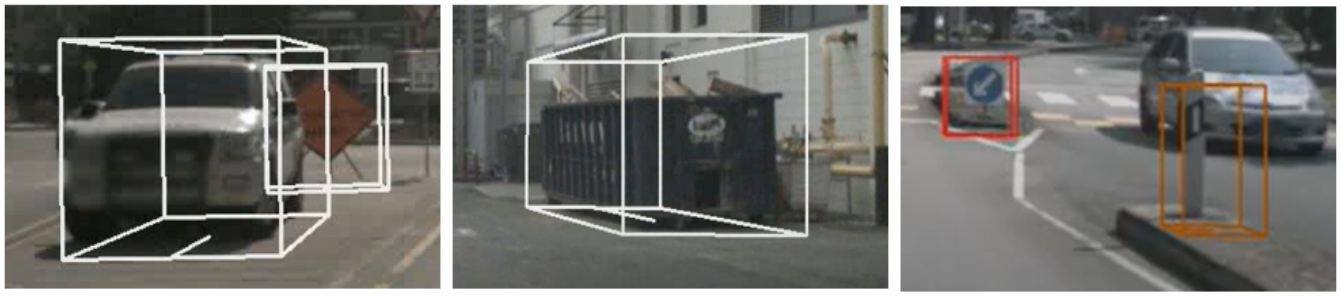


Fig. 1: *Motivating examples of "unknown" objects in the nuScenes dataset*. **Left**: `emergency vehicles` are annotated but never used in either training or testing 3D MOT systems, likely due to the sparsity of such examples (cf. Fig. 3). Therefore, a system trained on nuScenes closed-world data will either ignore, or misidentify this unknown object. **Center**: `debris` is annotated in nuScences, but like `emergency vehicles`, it is also not used in either training or testing. `debris` is itself a super-class that contains different objects of various shapes and sizes. **Right**: By manual inspection, we find other important objects like a `road sign` that are not annotated. Such objects are important for driving safety but too sparsely found to be annotated. Through qualitative visualization, we demonstrate that our method can detect and track some "true unknowns" (these are detection/tracking results from our model).

world MOT, we collectively group these hitherto ignored classes into a single open-world *unknown* super-class. We take inspiration from SOTA MOT methods and train using closed-world (i.e., known) classes and a small number of unknown objects necessary to define an "unknown" object catchall, but we evaluate the trained MOT model on testing examples from both *closed-world (i.e., known)* classes and unseen objects from the *open-world (i.e., unknown) superclass*. Therefore, we can naturally adopt the established evaluation protocols for closed-world data, and extend them with care to the open-world super-class.

II. Related Works

Open-World Perception. There are multiple lines of work addressing open-world perception, such as anomaly detection [9], [11], out-of-distribution detection [12], outlier detection [13], and open-set recognition [14]. These open-world problems can be crisply extended from closed-world image classification by evaluating a model in terms of its ability to reject outliers or unknown testing examples. Open-world image classification problems are typically explored in synthetic setups such as classification on the small-scale datasets (MNIST or CIFAR [10], [12]). In contrast, it is non-trivial to study open-world detection and open-world tracking, both of which are important modules in the autonomy stack. One exception is [2] that explores LiDAR-based open-set instance segmentation by simply learning a class-agnostic grouping model. However, [2] simplifies the study and only evaluates on two super-classes, *known* and *unknown*, without explicitly studying how the trained model performs on crucial closed-world classes. In literature, it appears that instance grouping methods do not perform as well as the class-aware detection methods [15], [16]. In contrast, we study open-world MOT and establish a more realistic setup where we consider performance on both closed-world known classes as well as open-world unknown classes, and show our method achieves SOTA performance on the closed-world with a strongly improved ability to detect and track unknown objects.

3D MOT for Autonomous Driving. 3D object detection and tracking are core vision problems and form important modules in autonomous driving. These two problems can be elegantly explored in the context of 3D MOT, which has been greatly advanced thanks to recent large-scale datasets [3], [6], [7]. In these datasets, LiDAR point clouds are a common input format that gives accurate 3D measurements which are crucial to an autonomous vehicle driving in the 3D world.

However, LiDAR data is sparse, which make it inefficient as input into the same 2D convolutional neural networks (CNN) that show great success on images [17], [18]. To efficiently learn from LiDAR data, some methods propose learning-friendly representations that convert LiDAR data into pseudo-images (e.g., bird-eye-view (BEV) images [19], [20]) and voxel-grid tensors [19]), which naturally embrace the established 2D detectors (based on CNN) for 3D object detection. Others adopt 3D sparse convolutions directly on the LiDAR point clouds, and achieve state-of-the-art 3D detection performance [4], [5].

Tracking-by-detection, a popular MOT paradigm, directly leverages bounding boxes from a detector and performs temporal association. Accordingly, [21] extends SORT [22] for 3D MOT, establishing a natural baseline. The tracking-by-detection framework illustrates that better 3D MOT performance can be achieved by independently using better 3D detector [21] and better data aggregation methods [23]. While using better detectors and trackers can surely improve 3D MOT, learning an end-to-end model for joint detection and tracking has shown promising results [5], [24], [25]. CenterPoint [5] is an innovative track-by-detection mehod that factorizes 3D objects into locations and appearance (i.e., shape and size). It learns a 3D model that jointly localizes the points (that represent objects for detection) and predicting points' velocity (which is used for tracking). For 3D object detection, CenterPoint further learns to regress object attributes including size and orientation. As previously noted, CenterPoint, as well

as other 3D MOT methods, are limited to only several known classes and cannot detect and track objects from open-world other classes. We address open-world 3D MOT in this work.

III. 3D Open-World Detection and Tracking

MOT is comprised of two distinct modules: object detection and tracking. Typically, MOT is explored in the closed world, and SOTA methods unanimously train models over some fixed set of known classes. By design, a closed-world MOT model is class-aware such that it only fires on spatial locations when it identifies an object as one of its known classes. As a result, closed-world MOT methods cannot detect and track unknown objects from novel classes. As we study open-world MOT, we first address the challenge of designing a model that is able to detect and track unknown objects.

In this section, we first review a SOTA MOT method, CenterPoint [5], and explain why fails to detect unknown objects.[1] Next, we introduce a naive post-hoc approach that detects unknown classes by filtering the detection results from CenterPoint. We extend this idea of creating a catchall class by retraining CenterPoint with an additional regression head for unknown objects and show that we can improve upon SOTA MOT performance on closed-world classes, as well as detecting and tracking in the open world.

A. A SOTA Method for Closed-World MOT

CenterPoint [5] is a SOTA tracking-by-detection method for closed-world MOT. In contrast with other MOT methods that detect objects (by predicting their shape), CenterPoint factorizes objects into location and shape. Such a factorization allows for exclusively training a detector and tracker by treating objects as points on spatial locations, diminishing the effects caused by object appearance variations. For each detected point representing an object, CenterPoint learns to regress the corresponding object size, orientation and velocity (which are directly used as location offsets for tracking). This keypoint regression based method shows superior MOT performance over other MOT methods that are trained on objects with entangled location and appearance/shape [4], [21]. However, similar to other methods, CenterPoint learns a class-aware detector (and tracker), meaning that it searches candidate objects spatially for each of the defined closed-world classes. As a result, CenterPoint, as well as other closed-world MOT methods, is inherently unable to detect unknown objects, which are not listed in any known classes used for training. We start by exploring open-world MOT with a naive modification to CenterPoint that allows us to localize *some* unknown objects, as elaborated below.

B. A Training-Free Baseline for Open-World MOT

We present a naive modification on the CenterPoint network towards a baseline for open-world MOT. We *simply threshold the detection results for each class class based on the confidence scores computed by the CenterPoint network and treat the filtered-out or low-confidence detection as unknown objects*. While this threholding-based method is simple, it has solid foundations from the following three lines of work.

In object detection, thresholding serves as a common post-processing step to filter out low-confidence results [18], [26]. Since low confidence results are often (but not always) false positives, thresholding tends to remove such errors [18]. It is worth noting the empirical success reveals that the *inductive bias* from closed-world classes causes the model to produce "false negatives" which look like objects from these classes [26]–[28]. For example, even though an object detection system might not explicitly model "emergency vehicles" as a dedicated class during training, its possible that a trained car detector might localize emergency vehicles. In this way, thresholding false positive "errors" in detection can be used to increase performance on both the car and unknown object classes. We highlight that CenterPoint's outputs inevitably contain MOT results for some unknown objects. This thresholding method serves as a viable way to localize unknown objects without retraining.

In open-world recognition, the confidence scores can largely tell whether a testing example is known or unknown [29]–[31]. In particular, a rather simple method for open-world recognition is just to threshold the classification softmax scores, treating the low-confident ones as open-world/unknown examples [30]. Therefore, the filtered-out results from CenterPoint reasonably contain MOT results for unknown objects (as well as for the true negatives, e.g., background regions).

We find an appropriate threshold through a per class parameter sweep on our small validation set as described in Section V

C. A Stronger Baseline for Open-World MOT

We explore a straightforward extension of the the CenterPoint network by adding an additional regression head to directly identify unknown objects. We freeze the weights of the SOTA CenterPoint backbone, as well as the closed-world class detector heads and train an unknown object branch from scratch. Despite the simplicity of the threshold approach, an explicit regression head for unknown objects helps to indirectly regularize the closed-world regression heads from falsely detecting and classifying annotated unknown bounding boxes in the training set. We can think of this baseline as defining a non-linear learnable threshold that separates known-objects from unknown-objects. Through experimental validation, we find that this simple modification slightly improves performance on closed-world classes, while also providing a substantial boost in performance for unknown objects.

Importantly, this baseline defines a concrete decision boundary for unknown objects, and broadens the vocabulary of the detector, better generalizing to true unknown objects. Our proposed method leverages both the stronger baseline and thresholding trick to further improve tracking performance.

IV. Evaluation Protocols

Evaluating MOT in the open-world is non-trivial due to the complex definition of an open-world object. In particular,

[1]Note that our method is model-agnostic that can work with other SOTA tracking-by-detection methods. We use CenterPoint in this work for its SOTA performance.

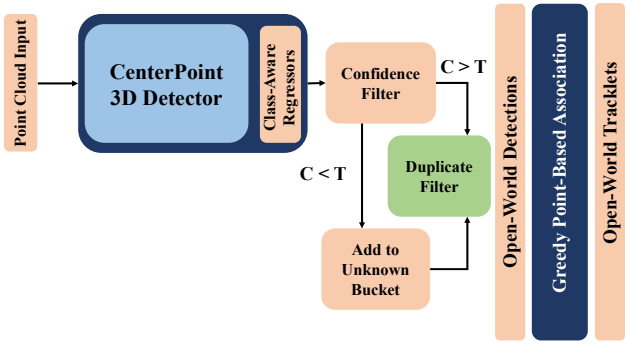


Fig. 2: Our proposed model is able to improve performance on closed world classes, while also improving robustness on open world previously unseen objects.

nuScenes only annotates a select number of unknown classes, making it difficult to evaluate open-world performance exactly as other known classes. As seen in Figure 1, there are also a number of unlabeled unknown objects of interest. We adapt the nuScenes evaluation protocol for evaluating open-world MOT. We first briefly describe the pertinent metrics used in the standard evaluation protocol, and then introduce modifications for the open world setting. Primarily, we focus on evaluating recall for open-world classes, since penalizing false positives according to the limited open-world annotations overlooks the unlabeled unknown instances.

A. Standard Evaluation Protocol

The standard nuScenes evaluation protocol evaluates detection and tracking performance on a fixed set of closed-world classes which models are explicitly trained on (C_{known}). Models are evaluated on these classes using various metrics which separately assess different aspects of object *detection* and *tracking* quality. We briefly introduce these metrics here, and refer the reader to [6] for further details.

For object detection, we report three metrics from the standard protocol: mean average precision (mAP), mean average translation error (mATE) and mean average orientation error (mAOE). The mAP metric is inspired by metrics for 2D object detection [32]. This metric first matches predicted objects for a class c with nearby groundtruth objects of the same class which have the smallest center-distance, up to a certain matching threshold. Given this matching, the metric computes the Average Precision (AP) at varying recall and matching thresholds for each class, and averages these across classes to compute the "mean average precision" (mAP). mATE and mAOE first compute the Euclidean distance (in meters) and the smallest yaw angle, respectively, between predicted objects and their corresponding matched groundtruth objects of the same class (at a matching threshold of 2 meters center-distance). These values are then averaged across recall thresholds $(10\%, 20\%, \dots 90\%)$ and classes. Importantly, "mATE" and "mAOE" are computed only for true positive predictions which are within the match threshold to a groundtruth bounding box, ignoring false positives, while "mAP" explicitly penalizes false positives.

For tracking, we report two additional metrics: Average Multi-Object Tracking Accuracy (AMOTA) and Identity F1 (ID-F1). AMOTA computes MOTA with a recall-normalization term, averaged across multiple recall thresholds. Specifically, for a single class, AMOTA is defined as

$$\text{AMOTA} = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \dots 1\}} \text{MOTAR}(r)$$

$$\text{MOTAR}(r) = \max\left(0, 1 - \frac{\text{IDS}_r + \text{FP}_r + \text{FN}_r - (1-r) * P}{r * P}\right),$$

where $\text{IDS}_r, \text{FN}_r, \text{FP}_r$ refers to the number of identity-switches, false-negatives and false-positives, r is the recall threshold varied over $n = 40$ different points, and P is the number of groundtruth objects. These 40 points are evenly space 2.5% apart between a recall of 2.5% and 100% inclusive [21]. If a MOT system is unable to reach a certain recall, the associated MOTAR score used in approximating the AMOTA integral is set to 0. This metric is computed per class, then averaged across all classes. AMOTA is conceptually similar to other integral metrics like average precision used for object detection.

ID-F1 [33] is not a part of the NuScenes protocol, but is a commonly used alternative metric to MOTA. ID-F1 more appropriately assesses tracking quality, while MOTA focuses primarily on detection quality. Intuitively, ID-F1 aims to evaluate trackers as instance-specific detectors: each predicted track id is assigned to a groundtruth track id, and evaluated as a detector for the assigned groundtruth object. Formally, ID-F1 is defined as

$$\text{ID-F1} = \frac{2IDTP}{2IDTP + IDFP + IDFN}$$

where IDTP (IDFN) is the number of groundtruth objects which are (not) correctly identified, while IDFP is the number of predicted objects which are correctly identified.

B. Extensions to the Open-World

Next, we introduce three extensions to the standard tracking metrics for open-world evaluation. Evaluating in the open-world is challenging, as the definition of a generic *object* is elusive and may be ambiguous [34]. For example, should an open-world detector treat each leaf on a tree as an individual object, or consider the entire tree as one object?

However, while designing a *precise* definition of objects (and by extension, non-objects) is challenging, it is easy to specify that certain regions of the world *are* objects, such as those in Fig. 1. To leverage this insight, we follow a rich line of prior work in 2D object detection [34], [35], focusing instead on the *recall* of unseen objects.

We supplement existing detection metrics in nuScenes by introducing the idea of mean average recall. Much like mAP, the mean average recall metric is inspired by 2D object detection [32]. This metric is similar to mAP, but does not consider false positives. This is ideal for our study of open-world classes since it is improper to penalize unlabeled "true unknown" objects. When evaluating the unknown class, we

restrict the number of predictions by only evaluating on the top K ($K = 100$) detections by confidence score.

In order to effectively evaluate open world tracking, we first define "MOTA recall in the open world" (MOTA_{ROW}), which adapts MOTA to measure only recall and identity-switches:

$$MOTA_{ROW} = 1 - \frac{FN + IDS}{N} \tag{1}$$

where, as before, FN is the number of false negatives and IDS is the number of identity switches. If we remove IDS, this reduces to classic recall.

Additionally, we use the ID-recall metric (IDR, [33]) as a supplement to ID-F1, which computes the portion of groundtruth tracklets which are correctly identified:

$$IDR = \frac{IDTP}{IDTP + IDFN} \tag{2}$$

We simplify evaluation by considering all unknown objects to belong to the same class. This allows the unknown class to be treated just like a closed-world object category.

C. Frame-wise Matching for MOT Evaluation on Unknowns

Directly evaluating on unknown object proposals will inevitably lead to poor performance due to an asymmetry between the number of annotated unknown objects and unlabeled unknown objects. Taking inspiration from single object tracking literature, we propose a first frame matching criterion to bridge the gap between closed-world and open world tracking. However, unlike single object tracking, we still detect and link frames together using a multi-object tracker. Our proposed matching algorithm has two criteria: (1) Every ground-truth object must be uniquely matched to a predicted tracklet, and both the ground-truth and predictions must be used at most once. This prevents multiple tracklets being generated for a single ground truth object. (2) We must match using only the first frame of the ground-truth. This approach to only using one frame of annotated ground-truth to bootstrap evaluation is well founded in literature, particularly in Video Object Tracking (VOT). Leveraging the first ground truth frame of an unknown object tracklet bootstraps the matching process and helps remove spurious tracklets. Additionally, using the first frame penalizes late detections (i.e. if the first instance that an object is annotated is at time t, we want to penalize trackers that only pick up this detection at time t + 10). A tracklet is not considered if it does not have a detection at the first frame of the ground truth. As a result, we indirectly bias towards early detections, which we argue is an important consideration to improve downstream task performance and overall system safety. Given these two criteria, we implement this first frame matching as shown in Algorithm 1.

Algorithm 1: Frame-wise Filtering Predictions for MOT Evaluation on Unknowns

Result: Set of IDs P corresponding to predicted tracklets that satisfy the maximal bipartite match.

$GT \leftarrow \emptyset, P \leftarrow \emptyset, M{<}K, V{>} \leftarrow \{\}$
// Collect first-frame of tracks.
for $t \in$ *timestamps* **do**
  Let Box_{GT} represent ground truth bounding boxes at time t, Box_P predicted bounding boxes at time t
  for $box \in Box_{GT}$ **do**
    if $box.ID_{tracklet} \notin GT$ **then**
      $GT \leftarrow box.ID_{tracklet} \cup GT$
      M[t].add(box)
    end
  end
end
for $t \in$ *M.keys()* **do**
  Let L_2 be a function that finds the pairwise center-distance between two sets of boxes.
  Affinity = L_2(M[t], Box_P[t])
  for *box in M[t]* **do**
    Let Proposals represent the set of boxes in Box_P[t] that are within m meters of box that are sorted in order of detection confidence.
    for $box \in$ *Proposals* **do**
      if $box.ID_{tracklet} \notin P$ **then**
        $P \leftarrow box.ID_{tracklet} \cup P$
        break
      end
    end
  end
end

D. nuScenes Dataset

We benchmark open-world MOT on the nuScenes dataset [6], a well-established and large-scale dataset that contains 3D annotations on LiDAR point clouds of diverse classes, and supports self-driving research on 3D detection and tracking.

Since the standard test protocol does not support open-world evaluation, we maintain the original train split, but devise a new validation and test split derived from the official validation set. These new splits partition the 150 scenes allocated to the traditional validation splits into 50 and 100 fixed scenes respectively for our open-world validation and test sets.

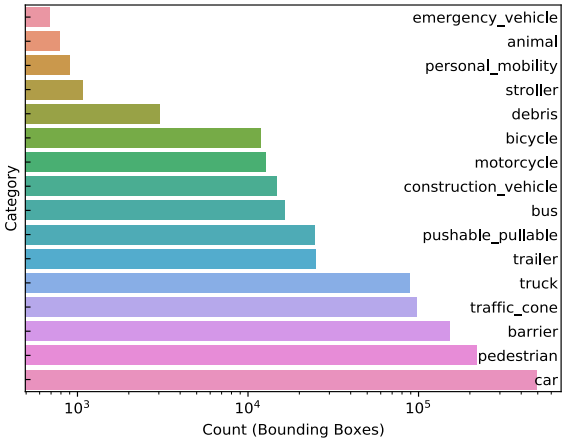
V. Experiments

We benchmark the effectiveness of our proposed open-world MOT pipeline through extensive experimentation on our proposed nuScenes validation and test splits.

A. Implementation

We implement our methods on top of the released CenterPoint code, written in PyTorch. Similarly, we modify the nuScenes evaluation toolkit to facilitate both open-world and closed-world evaluation. We use a pre-trained CenterPoint model in all of our models to ensure the they minimally maintain SOTA closed-world performance.

We train our stronger baseline model for 20 epochs using an Adam optimizer [36] and use a one-cycle learning rate



Closed-World Detection	Car, Pedestrian, Truck, Bicycle, Motorcycle, Trailer, Bus, Construction Vehicle, Barrier, Traffic Cone
Closed-World Tracking	Car, Pedestrian, Truck, Bicycle, Motorcycle, Trailer, Bus
Open-World MOT	Emergency Vehicles, Strollers, Personal Mobility, Debris, Pushable Pullable, Animal

Fig. 3: We plot the distribution of all annotated classes (left) in the nuScenes dataset. As with many real world datasets, the number of annotations per class follows the long-tail distribution. Classes traditionally used for closed-world detection and tracking (right) are well represented in the dataset. In this closed-world setting, MOT models are trained and evaluated on the same fixed set of classes. Current MOT tracking-by-detection pipelines are typically designed for the closed-world. It is important to note that closed-world tracking is interested in moving objects, and therefore does not evaluate barriers, traffic cones, or construction vehicles.

scheduler [37]. After training CenterPoint with the additional unknown object branch, we leverage the thresholding trick (as detailed in Fig. 2) to remove less-confident detected objects before generating detection results and tracklets.

B. Ablation Study

We study the effect of thresholding on the overall performance of our proposed open-world MOT model. We quantify our findings in Table I on both CenterPoint and our improved baseline.

Our thresholding strategy (Sec. III-B) slightly improves the AMOTA score of known classes for CenterPoint. However, we note that known classes perform slightly worse on $MOTA_{ROW}$, IDF_1, and IDR. Intuitively, as we reduce the number of predicted tracklets, recall based metrics are more likely to get lower. Unlike our initial prediction, simply thresholding low confidence predictions from known classes does not seem to impact unknown object performance. Both of these trends can be explained by the quality of low confidence detections. Although our initial hypothesis that low confidence detections from known classes correspond to false positives is substantiated by our experimental results, we find that these thresholded detections represent fragmented tracklets that cannot be properly linked by a tracker.

Our stronger baseline model that trains an extra unknown object regression head (Sec. III-C) performs slightly better on known classes, and significantly better on unknown objects. Interestingly, thresholding known class results from our stronger baseline model increases known class performance on all metrics, but significantly reduces unknown object performance. This suggests that when we explicitly train a regression head for unknown objects, we should leverage thresholding as a means to improve known class performance, but not use these thresholded results to supplement the unknown object. Training an unknown object regressor acts as a regularization method such that known class predictions are less likely to detect unknown objects, ensuring that thresholded predictions correspond to true false positives.

C. 3D Detection in the Open-World

Robust 3D detection is the backbone of MOT pipelines. In most cases, better detections result in better tracklet generation. Given the importance of 3D detection, we are interested in evaluating the performance of our proposed method against CenterPoint. Specifically, we are interested in determining the additional benefit provided by training on a small number of unknown objects, and using a per-class threshold to filter low confidence detections. We note that our proposed method has slightly higher performance on known classes, as expected. In Table II, we report the average translation error (ATE) and average orientation error (AOE), two key metrics in measuring the effectiveness of bounding box localization. We find that both ATE and AOE slightly improve for known objects, and additionally see significant improvement across all metrics for unknown object detection performance.

D. 3D Tracking in the Open-World

After detecting open-world unknown objects, tracking follows as a natural extension. Utilizing the same baseline and proposed models as in the previous section, we evaluate their performance on both closed-world and open-world tracking metrics. Specifically, we evaluate unknown objects using our first-frame matching protocol as defined in Algorithm 1. Table III highlights the per-class performance of both our models. Impressively, we find that training an additional unknown object branch without modifying the weights for the backbone

Known / Unknown	$AMOTA \uparrow$	$MOTA_{ROW} \uparrow$	$IDF_1 \uparrow$	$IDR \uparrow$
CenterPoint	0.604 / 0	0.671 / 0	0.713 / 0	0.659 / 0
CenterPoint + Threshold	0.609 / 0	0.652 / 0	0.708 / 0	0.642 / 0
CenterPoint + Unknown	0.605 / 0.476	0.657 / 0.564	0.707 / 0.690	0.648 / 0.564
CenterPoint + Unknown + Threshold	0.608 / 0.437	0.661 / 0.518	0.712 / 0.656	0.651 / 0.518

TABLE I: Summary of different baseline models to evaluate the effectiveness of thresholding on the open world test split. We average performance over the 10 closed-world classes to report known-object performance, and separately report performance on the unknown super-class. Training an additional regression head improves unknown object performance, while thresholding improves known class performance.

Model	Class	$mAP \uparrow$	$mAR \uparrow$	$mATE \downarrow$	$mAOE \downarrow$
CenterPoint	Bicycle	0.315	0.814	0.178	0.381
	Bus	0.727	0.810	0.278	0.023
	Car	0.855	0.923	0.174	0.107
	Motorcycle	0.576	0.847	0.195	0.252
	Pedestrian	0.848	0.962	0.144	0.384
	Trailer	0.389	0.615	0.495	0.390
	Truck	0.566	0.809	0.311	0.052
	Barrier	0.672	0.885	0.189	0.060
	Construction Vehicle	0.216	0.504	0.706	0.953
	Traffic Cone	0.687	0.926	0.131	N/A
	Known Objects	0.585	0.809	0.280	0.289
	Unknown Objects	0	0	1	1
	Summary	0.532	0.736	0.346	0.360
CenterPoint + Unknown + Threshold	Bicycle	0.312	0.805	0.177	0.362
	Bus	0.727	0.806	0.276	0.018
	Car	0.852	0.915	0.172	0.105
	Motorcycle	0.576	0.818	0.192	0.255
	Pedestrian	0.844	0.951	0.143	0.382
	Trailer	0.389	0.604	0.493	0.397
	Truck	0.569	0.809	0.312	0.053
	Barrier	0.671	0.879	0.188	0.061
	Construction Vehicle	0.214	0.5	0.709	0.952
	Traffic Cone	0.686	0.914	0.129	N/A
	Known Objects	0.584	0.800	0.279	0.287
	Unknown Objects	0.119	0.885	0.230	0.627
	Summary	0.542	0.808	0.275	0.321

TABLE II: We compare our proposed open-world detection performance with CenterPoint, a SOTA MOT method on our open world test split. Our proposed method significantly improves on all metrics for unknown objects while maintaining closed-world performance.

network indirectly improves known object AMOTA for most classes.

VI. Conclusion

In this paper, we have extensively explored the task of open-world MOT, and have established simple yet effective ways of endowing SOTA closed-world MOT systems with open-world robustness. For future work, we will train a two-stream model that provide SOTA closed-world performance by one stream and unknown MOT by the other. Additionally, we will annotate some real unknown objects (e.g., temporary signs) as test-set to gauge how our model can detect and track these crucial unknown objects.

Acknowledgements

Neehar Peri was supported in part by the National Science Foundation CISE REU Grant #1659774 and the CMU Argo AI Center for Autonomous Vehicle Research.

Model	Class	$AMOTA\uparrow$	$MOTA_{ROW}\uparrow$	$IDF_1\uparrow$	$IDR\uparrow$
CenterPoint	Bicycle	0.356	0.519	0.589	0.515
	Bus	0.835	0.822	0.872	0.822
	Car	0.839	0.844	0.839	0.828
	Motorcycle	0.559	0.603	0.685	0.589
	Pedestrian	0.765	0.815	0.786	0.794
	Trailer	0.486	0.555	0.629	0.528
	Truck	0.643	0.703	0.743	0.697
	Barrier	0.627	0.692	0.720	0.681
	Construction Vehicle	0.323	0.473	0.553	0.457
	Traffic Cone	0.614	0.691	0.722	0.683
	Known Objects	0.605	0.672	0.714	0.660
	Unknown Objects	0	0	0	0
	Summary	0.550	0.611	0.649	0.600
CenterPoint + Unknown + Threshold	Bicycle	0.329	0.476	0.566	0.470
	Bus	0.833	0.818	0.871	0.819
	Car	0.842	0.819	0.841	0.807
	Motorcycle	0.608	0.652	0.722	0.643
	Pedestrian	0.776	0.805	0.791	0.785
	Trailer	0.495	0.480	0.606	0.469
	Truck	0.648	0.728	0.751	0.329
	Barrier	0.635	0.689	0.720	0.678
	Construction Vehicle	0.306	0.433	0.523	0.417
	Traffic Cone	0.615	0.713	0.733	0.706
	Known Objects	0.609	0.661	0.712	0.651
	Unknown Objects	0.437	0.518	0.656	0.518
	Summary	0.593	0.648	0.707	0.639

TABLE III: We compare our proposed open-world tracking performance with CenterPoint, a SOTA MOT method on our open world test split. Our proposed method slightly improves AMOTA on known classes and significantly improves open world robustness.

REFERENCES

[1] A. Taeihagh and H. S. M. Lim, "Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transport Reviews*, vol. 39, no. 1, pp. 103–128, 2019.

[2] K. Wong, S. Wang, M. Ren, M. Liang, and R. Urtasun, "Identifying unknown instances for autonomous driving," in *CoRL*, 2020.

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[4] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3d object detection," *arXiv preprint arXiv:1908.09492*, 2019.

[5] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," *arXiv preprint arXiv:2006.11275*, 2020.

[6] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[7] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[8] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *CVPR*, 2019.

[9] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," in *ICLR*, 2019.

[10] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *NeurIPS*, 2018.

[11] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.

[12] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *ICLR*, 2018.

[13] S. Pidhorskyi, R. Almohsen, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," in *NeurIPS*, 2018.

[14] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *TPAMI*, vol. 35, no. 7, pp. 1757–1772, 2012.

[15] S. Kong and C. C. Fowlkes, "Recurrent scene parsing with perspective understanding in the loop," in *CVPR*, 2018.

[16] H. Wang, R. Luo, M. Maire, and G. Shakhnarovich, "Pixel consensus voting for panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9464–9473.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeuriPS*, 2012.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017.

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 12 697–12 705.

[20] P. Hu, J. Ziglar, D. Held, and D. Ramanan, "What you see is what you get: Exploiting visibility for 3d object detection," *CoRR*, vol. abs/1912.04986, 2019.

[21] X. Weng and K. Kitani, "A baseline for 3d multi-object tracking," *arXiv preprint arXiv:1907.03961*, 2019.

[22] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing, ICIP*. IEEE, 2016, pp. 3464–3468.

[23] H. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3d multi-object tracking for autonomous driving," *CoRR*, vol. abs/2001.05673, 2020.
[24] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 3569–3577.
[25] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun, "Pnpnet: End-to-end perception and prediction with tracking in the loop," *CoRR*, 2020.
[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
[27] K.-Y. Chang, T.-L. Liu, H.-T. Chen, and S.-H. Lai, "Fusing generic objectness and visual saliency for salient object detection," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 914–921.
[28] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3286–3293.
[29] A. Bendale and T. E. Boult, "Towards open set deep networks," in *CVPR*, 2016.
[30] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *ICLR*, 2017.
[31] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky, "Your classifier is secretly an energy based model and you should treat it like one," in *ICLR*, 2019.
[32] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Computer Vision - ECCV 2014*, ser. Lecture Notes in Computer Science, vol. 8693. Springer, 2014, pp. 740–755.
[33] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Computer Vision - ECCV 2016 Workshops*, ser. Lecture Notes in Computer Science, G. Hua and H. Jégou, Eds., vol. 9914, 2016, pp. 17–35.
[34] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2189–2202, 2012.
[35] P. H. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *Advances in Neural Information Processing Systems*, 2015, pp. 1990–1998.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*, Y. Bengio and Y. LeCun, Eds., 2015.
[37] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision, WACV*. IEEE Computer Society, 2017, pp. 464–472.

Dynamic Differentiation between Leading Indicators of Hemorrhagic and Septic Shock

Ernest Pokropek[1], Xinyu Li[2], Michael R. Pinsky[3] and Artur Dubrawski[2]

***Abstract*— Distinction between hemorrhage and sepsis still poses a challenge in early diagnosis, as both medical conditions share drop of the blood pressure, and consequently hypotension as one of the initial symptoms. What is crucial, they require vastly different medical treatments; if timely therapy or resuscitation is missing, both conditions could lead to adverse patient outcomes, or even death. Currently, distinguishing between the two highly depends on the qualifications and promptness of the medical staff, making it challenging even in controlled environments. We utilize a Random Forest Classifier model capable of differentiating between hemorrhage and endotoxin-induced sepsis at both early and late stages of decreasing blood pressure, based on arterial pressure measurements. Both the training and the evaluation of our classifier is conducted on different stages of the aforementioned phenomena, using laboratory data collected from pigs exposed to bleeding or endotoxin infusion.**

***Index Terms*— Machine learning, Medical diagnosis, Decision support systems**

I. INTRODUCTION

Hemorrhage is the process of blood escaping the circulatory system via damaged blood vessels, either outside of the body (external) or inside it (internal). In the first case, detection of this condition might seem straightforward, due to factors such as visible wounds or the blood itself, yet for the latter, the diagnosis becomes more complicated, with necessity of higher clinical suspicion. This includes various methods of evaluation of the presumptions related to bleeding including CT, X-ray, or ultrasound scans [1], of which selection highly depends on the suspected placement of the traumatic wound. The treatment varies on factors such as anatomic location and extent of the injury, including methods such as resuscitation and transfusion. Where most of adults could tolerate loss of blood up to 14% of its total volume, without prompt diagnosis it may later lead to organ failure, coma, and death. Hemorrhage, although preventable, when untreated might be fatal [2], with approximately 2 million deaths annually (worldwide) as a result of this medical condition [3].

Most potent trigger of sepsis are endotoxins - lipopolysaccharides (LPS), which contribute to the major part of the outer membrane of gram-negative bacteria. Their presence in the blood, caused by the bacterial infection, might lead to septic reactions and even end in death of the patient due to endotoxic shock [4]. Sepsis is usually diagnosed by presence of endotoxin in the blood stream, and it is most commonly performed using the different variations of Limulus Amoebocyte Lysate (LAL) assay. This technique utilizes the protein from blood of the horseshoe crab, *Limulus polyphemus*, which makes it clot in the presence of LPS. This method requires taking a blood sample from the patient in order to perform the diagnosis. Rapid treatment here is of extreme importance, given that some of the infecting bacteria might be resistant to antibiotics. This is especially present in Healthcare-Acquired Infections (HAIs), often involving life-threatening gram-negative bacteria such as *Escherichia coli* or *Pseudomonas aeruginosa*. HAIs are responsible for approximately 1.7 million deaths each year in the United States alone [5].

Both of the aforementioned phenomena share one of the initial symptoms, which is hypotension [2], [6]. It may be caused by hypovolemia (reduced blood volume) which occurs when exposed to hemorrhage or sepsis (which most commonly is triggered by LPS) [7]. Both medical conditions may lead to development of lethal inflammatory responses or life-threatening systemic inflammatory response syndrome (SIRS) capable of irreversible end-organ dysfunction [8]. Given an automated method for distinction between these two medical conditions based on blood pressure alone, it would drastically improve the medical resilience for quick and proper treatment, especially when there is no possibility to inquire about patient's recent history. This could also serve as a great support for current methods of distinction, as they are usually costly and time consuming. This approach would utilize already available data, given that some of the patients are constantly monitored for vital signs. In this study, we develop a Random Forest Classifier model that is able to distinguish between the two aforementioned medical conditions in different time intervals of their presence, based on the arterial pressure (describing the pressure exerted by the blood in large arteries).

II. PRIOR WORK

With the enormous amounts of data being collected from the patients, including the vital signs, their interpretation is becoming more troubling for human eye. This is an excellent opportunity to apply the machine learning (ML) models, which can quickly estimate, or act as a support of the medical diagnosis. ML has been successfully used for various tasks addressing this opportunity, including cardiovascular diseases or cancer [9].

[1]Ernest Pokropek is with Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland ernest.pokropek.stud@pw.edu.pl

[2]Xinyu Li and Artur Dubrawski are with the Auton Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA xinyul2@andrew.cmu.edu, awd@cs.cmu.edu

[3]Michael R. Pinsky is with Department of Critical Care Medicine, University of Pittsburgh, Pittsburgh, PA, USA pinsky@pitt.edu

Application of ML in the medical diagnosis may vary from using computer vision to analyze the MRI scans to multi-dimensional regression models examining the raw vital signs. ML approach has been also used for detection of hemorrhage: Kuo et al. [10] proposed a convolutional neural network being able to detect different types of intracranial hemorrhage based on CT scans of human head. Their approach outperformed some of the professional radiologists in accuracy of detecting the aforementioned medical condition. Furthermore, Nagpal et al. [11] presented a model being able to dynamically detect hemorrhage by using waveform of Central Venous Pressure (CVP), outperforming previous ML approaches addressing this problem in terms of speed of detection and providing better foundations for further evaluation of patient's health for clinicians.

There is little prior work related to automated and dynamic detection of endotoxin in blood due to the necessity of performing exhaustive blood tests for majority of current methods. However, one of the most important consequences of LPS presence in blood is sepsis, and consequently septic shock. Detection of this critical condition has been successfully performed by utilizing long short-term memory neural network in [12]. This model is able to detect majority of potential septic shock onsets about 40 hours prior to them, basing on not only vital signs, but also laboratory tests, medical procedures, medications, diagnoses and demographics. For the sepsis alone, Lauritsen et al. [13] proposed different ML models for detection of this medical condition based on sequential, time-stamped data including diagnoses, procedures, treatments, various medical image data such as CT, X-ray, ultrasound scans, and demographics. This approach yielded an ability to detect sepsis onset 24 hours prior to it, with substantial part of diagnoses performed correctly. All those methods are proper and well thought, yet do not provide a dynamic detection necessary for prompt diagnosis basing only on the vital signs, which is especially important for intensive care patients that have just arrived to the medical facility. Henry et al. [14] presented an automated method of detecting potential development of septic shock. Based on real-time data, their model is able to detect this medical condition approximately 28.2 hours prior to its onset.

There exist traditional methods for distinction between hemorrhage and sepsis (or its trigger, i.e. endotoxin), but they are usually limited to methodology of eliminating possibilities. For example, during septic shock, major blood loss from gastrointestinal hemorrhage is uncommon [15], thus when facing such phenomena, our intuition may lean towards diagnosis of internal bleeding. Currently, to our knowledge, there is no popular and reliable method for dynamic and automated distinction between hemorrhage and sepsis.

III. DATA AND METHODOLOGY

For this study we used a real-world laboratory data collected from 23 healthy Yorkshire Pigs. Animals were divided into two groups: exposed to bleeding (15 individuals, hemorrhage group) and infused with LPS (8 individuals, endotoxin group). For each of the two classes, after setting up necessary measurement devices, pigs were undergoing the *stabilization phase*, which lasted approximately 50 minutes. During this episode, the subjects were not exposed to bleeding nor LPS infusion. After the stabilization period, the pigs from the first group were exposed to slow bleeding (5 milliliters per minute), and the pigs from the second group - to the first injection of LPS. This injection for the group exposed to endotoxin has been performed in order to injure the subjects immune system enough to develop septic shock symptoms after the second infusion, which is the subject of our interest in this study. During the experiment, the waveform of arterial pressure has been collected at 250Hz pace. Average length of the examined episodes are approximately 68 minutes for the hemorrhage group (minimum 50, maximum 78, standard deviation of approximately 8 minutes) and 58 minutes for the endotoxin group (minimum 45, maximum 115, standard deviation of approximately 23 minutes).

A. Pre-processing and Featurization

For every trailing time window of 4 minutes length, statistical figures of arterial pressure have been computed: mean, median, standard deviation and range between 5th and 95th percentile. Together with raw measurement, those features contribute to one point of the featurized data per one 4 minutes window. Those features have been later downsampled to 2Hz along with computing the Discrete Fourier Transform (DFT) to calculate signal's power for different frequency bands, which could be interpreted in medical diagnosis. Following frequency bands have been investigated, using 250Hz sampling frequency:

- $[0.04, 0.15)$ - Low Frequency (LF)
- $[0.15, 0.4)$ - High Frequency (HF): corresponds to respiratory rate
- $[0.4, 10)$ - Very High Frequency (VHF): corresponds to heart rate
- $[10, 125)$ - Very-Very High Frequency (VVHF)

Furthermore, the data for each pig has been standardized using its stabilization phase. Given i-th feature of arterial pressure during stabilization, X_i, the corresponding standardized feature $\hat{X}_i$ may be calculated as shown in 1.

$$\hat{X}_i = \frac{X_i - \widetilde{X}_{s_i}}{IQR(X_{s_i})} \tag{1}$$

where:

$\hat{X}_i$ = i-th standardized feature
X_i = original i-th feature
X_{s_i} = i-th feature from stabilization phase
$\widetilde{X}_{s_i}$ = median of the i-th feature from stabilization phase
IQR = interquartile range,

Finally, the last task of the data pre-processing was to overcome the problem related to episode selection. As for the bleeding group we used the episode immediately following the stabilization phase, for group exposed to endotoxin we had to skip the aftermath of first LPS infusion as its purpose was to only destroy the subject's immune system, rather

than develop actual septic symptoms. This resulted in the waveform not being continuous, with more or less noticeable change from the stabilization phase to the second episode of LPS infusion. This change on the boundary between episodes has mean relative magnitude (with respect to the last sample of the stabilization phase) of 6.22%, with standard deviation of 4.80%. This problem is illustrated on 1.

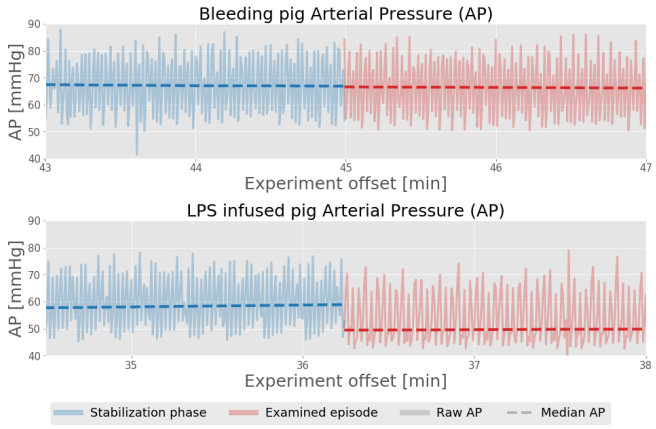


Fig. 1. Differences in transition from stabilization phase to examined episode (first hemorrhage episode for bleeding group, second infusion of LPS for endotoxin group) pictured using data of two individual pigs.

We overcame this difficulty by truncating the data used later for training to fall in the time interval of [25, 40] minutes of offset (absolute time from the beginning of the experiment). The first few minutes correspond to beginning of the experiment, where pigs are either exposed to bleeding, or to second infusion of LPS. Due to the difference in transition, we decided to skip this part of the episode in order not to bias the model with possible outliers that come from problems related to improper stabilization or individual differences. Similarly, we chose the right bound of the interval, which is 40-th minute: after such time, the vital sign tend to diminish, so the model might be more exposed to individual differences between the groups. This interval yielded in highest performance scores across all whole data set. The data for testing remains unchanged for all the analyses.

B. Experiment setup

We used the Random Forest Classifier (RFC) model with 80 Decision Trees, where each tree had maximum depth limited to 30. For proper performance analysis, we performed a specific utilization of k-Fold cross validation: first, the pigs from each group were joined into unique pairs (1 pig from bleeding group together with 1 pig from endotoxin group), which resulted in total of 120 such pairs. Then, for each run of the training, one of the pairs has been used for testing the performance, resulting in total of 15 * 8 = 120 folds of cross validation, i.e. train-test runs. Although the training data has been truncated, as described as in III-A, the testing has been performed on the whole volume of the data, including the early and late stages of the LPS infusion. In this binary classification task, the condition of endotoxin-induced sepsis was marked as the positive class (1), and the hemorrhage as negative (0).

IV. RESULTS

The presented results are in form of averaged test results from the 120 folds of cross validation, where in each fold the data used for testing have not been influenced by any process except normalization as in eq. 1. Each fold represents training and evaluating the model's performance on unique pair of subjects, which consists of two animals from different groups (hemorrhage, endotoxin).

A. Performance

First metric to evaluate the model, is the Receiver Operating Characteristic (ROC) curve. This plot, having the True Positive Rate (TPR) on the y-axis, and False Positive Rate (FPR) on the x-axis describes the trade-off between accuracy and sensitivity of the model, which especially in medical diagnosis, is crucial for proper analysis. Figure 3, presents classifier's ability to distinguish between endotoxin and hemorrhage for the model, which at the lowest False Positive Ratio (FPR) can detect on average over 60% of endotoxin (positive class) occurrences. This provides appropriate metric for medical diagnosis, as this corresponds to 60% rate of detection without making a single mistake. The standard deviation of the ROCs across all the pairs from cross validation is constantly above the random line, which implies that on average, for each test run the model has been better than random prediction.

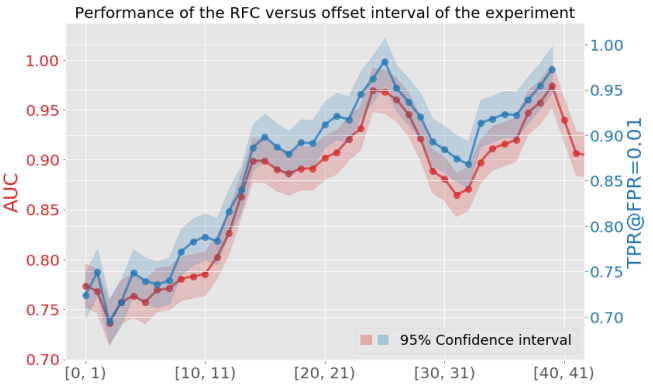


Fig. 2. Comparison of average AUC (left side y-axis, red color) and True Positive Ratio at constant False Positive Ratio equal to 0.1 (right side y-axis, blue color) for given intervals of time offset. Each of the intervals are left-side inclusive and right-side exclusive. Scores are computed on the testing data at each fold, and then averaged. Data used for testing have not been truncated.

Even with a truncated data with respect to the [25, 40] minute time interval, the RFC model performs reasonably well on the beginning of the experiment, with increasing trend towards the later moments. This is illustrated on 2, where the average AUC (Area Under (ROC) Curve) and TPR at FPR=0.01 are calculated for each minute of the experiment. Intuitively, the RFC model is confused at the beginning, where both of the medical conditions have just started

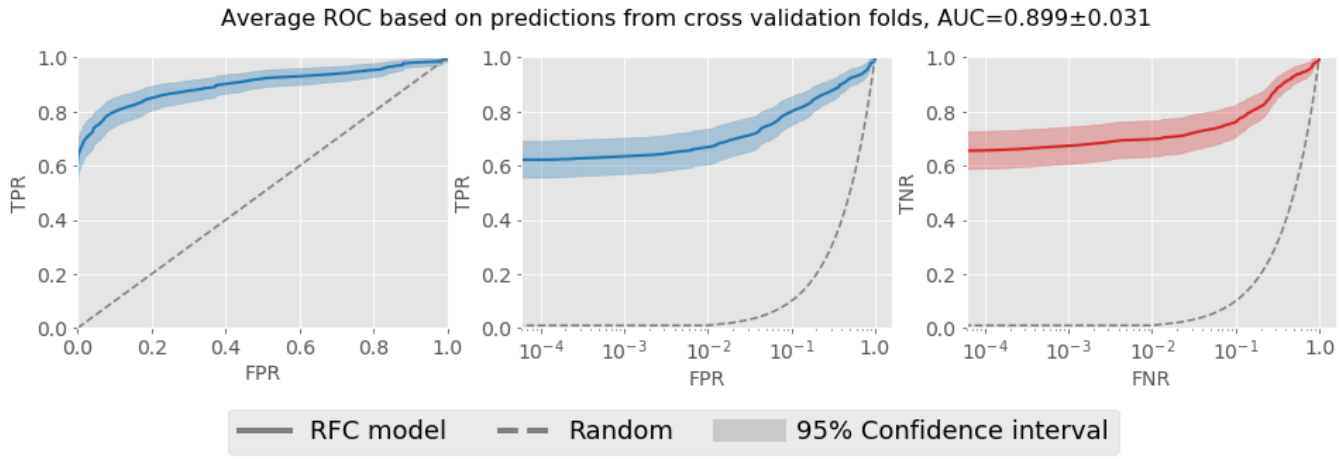


Fig. 3. ROC curves calculated in each fold of the cross validation, based on two pigs from opposite group and then averaged (after interpolating for all possible FPR values). Positive class corresponds to LPS infused group (endotoxin) and negative for animals exposed to bleeding (hemorrhage). Second plot presents the same data as first one, yet it has x-axis in the logarythmic scale. The third plot pictures True Negative Ratio (TNR) to False Negative Ratio (FNR). The confidence of average AUC has been calculated using 95% confidence interval ($z = 1.96$).

to set in. With slower changes and healthier physiology of the subject, the distinction between two life-threatening conditions becomes more difficult. With progress of time, the predictions are more confident and correct, just to reach and optimal decisive properties around 25th minute of the progressing medical condition. This analysis demonstrates that our solution for distinction between hemorrhage and endotoxin could serve as a really useful support for medical diagnosis even on the very first stages of aforementioned medical conditions.

Lastly, a cumulative confusion matrix has been calculated by calculating it based on predictions made at each of 120 folds of cross-validation. The calculated metrics obtained from it are presented in I, where: TP - True Positive, TN - True Negative, FP - False Positive, FN - False Negative.

Furthermore, the cumulative confusion matrix is presented graphically on 4.

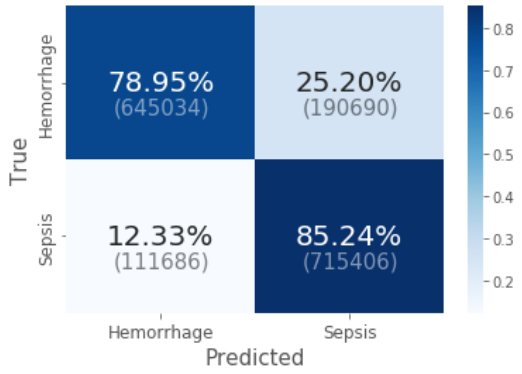


Fig. 4. Cumulative confusion matrix based on predictions of each fold of the cross validation. The percentages correspond to the portion of classified samples in respect to total number of values in each of the examined groups (hemorrhage, endotoxin) and the values in parenthesis to the number of predictions.

Metric	Definition	Value
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	81.82%
F-1 Score	$\frac{2TP}{2TP+FP+FN}$	81.01%
Sensitivity	$\frac{TP}{TP+FN}$	85.24%
Specificity	$\frac{TN}{TN+FP}$	78.95%
Precision	$\frac{TP}{TP+FP}$	77.18%
Miss Rate	$\frac{FN}{FN+TP}$	14.76%
Fall-out	$\frac{FP}{FP+TN}$	21.05%
False Discovery Rate	$\frac{FP}{FP+TP}$	22.82%
False Omission Rate	$\frac{FN}{FN+TN}$	13.50%

TABLE I
AVERAGE METRICS OBTAINED FROM THE CUMULATIVE CONFUSION MATRIX SHOWN.

The tendency of model classifying rather the negative class (bleeding) comes from the fact, that there are 15 pigs exposed to hemorrhage, compared to only 8 infused with LPS - that is why the the sensitivity (recall) has smaller magnitude than precision. In general, the model has the right tendency when making decisions, with majority (82.06%) of samples classified correctly. It is worth to emphasize, that those scores come from the whole duration of the examined episode, and they are influenced by the poorer performance of the models at the beginning.

Although the general performance of the model is important, in medical diagnosis for critical care we are especially

interested in low false alarm ratio. Sometimes we may want our model to classify even only 20% of the insets of given medical condition while having a really small False Positive Ratio, than have a model performing random guesses and thus detecting around 50% of the target. To illustrate the dexterity of our solution with respect to aforementioned reasoning, Fig. 5 presents the confusion matrix calculated for probability threshold of 0.01 (if the model predicted probability larger or equal to 0.01 for certain sample it got classified as positive class, otherwise as negative). For such tuning the model makes only about 3 mistakes per each 100 decisions when detecting sepsis, given that it makes 2 predictions per second.

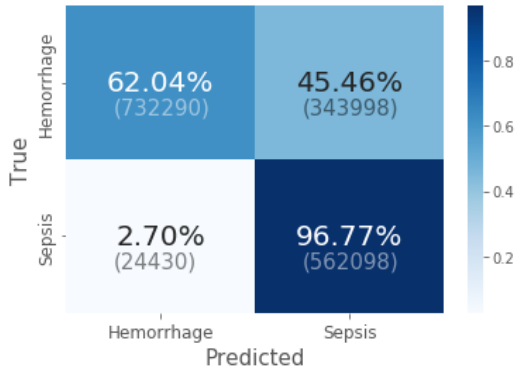


Fig. 5. Cumulative confusion matrix with probability threshold equal to 0.1. Percentages correspond to the the portion of whole class population that has been detected.

B. Decision making

To have an insight on how the model is making decisions, we analyzed the feature importance of trained RFCs. Importance of given feature is defined as how much it contributes to the final prediction, i.e. how important it is. Here, we considered attributes with importance higher than 8% to be informative, that is that they have a notable influence on the prediction. Figure 6 shows, that the most important feature is the one corresponding to respiratory rate of the subject (28.99%), which is one of the most important vital signs. Consequently, very important is also the one corresponding to heart rate (28.10%). This is quite intuitive and expected, as the arterial pressure is directly linked to this vital sign. The standard deviation (16.52%) resembles the dynamics and scale of changes of the subjects anatomical responses, which are directly referring to how stable the actual patient is - with more prompt changes, that is more rapid response to anatomical state, we may expect the patient to be healthier, i.e. the bleeding or sepsis have not influenced it much yet. With notable influence on the decision, there are as well features which represent the magnitude of the arterial pressure itself.

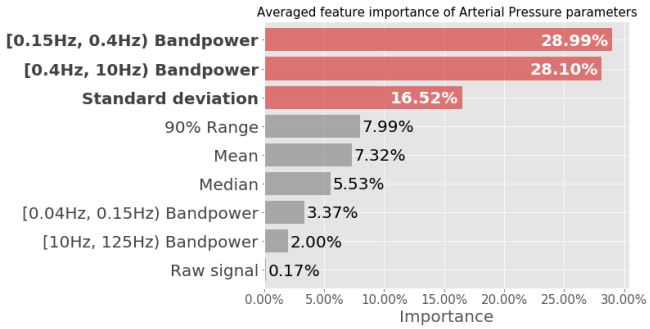


Fig. 6. Average feature importances from 120 models trained during the cross-validation, where informative features are marked with red colour.

V. DISCUSSION

Measurement of arterial pressure is one of the most invasive methods for vital sign monitoring. At the beginning of the research, we wanted to incorporate Pulse Oximeter Pleth (Plethysmograph) into the feature set as well, with same methodology as described as in III-A. Together with calculated Pleth Variability Index (PVI), it could provide a great opportunity for distinction between hemorrhage and endotoxin for patients which state is not critical, as it is a non-invasive method of measuring the hearth rhythm. This method of measuring the vital signs is also easier to set up. However, there is a trade-off between simplicity, easiness of use, and precision. The problem we faced, that compared to arterial pressure recordings, Pleth had incredible high variability between the groups - based on only median of the Pleth signal, the model was able to distinguish between the two medical conditions with incredible confidence. After a throughout distribution analysis, measuring performance for different offsets and thresholds of variables, we discovered that the Pleth signal brought too much differences based on the selected groups, instead of the medical condition, thus it would be useless in real world applications. Due to this fact, we decided to use only the arterial pressure for this study. For future work and potential development, different vital sign measurements might be taken into account, with respect to their invasiveness.

It is worth to mention, that the data set is quite small in terms of individuals - having only 25 different animals, it is quite challenging to provide a model that will perform well on whole population. Future work might involve analysis of larger groups, with more focus put into diminishing the problems related to individual differences. The next steps of this study should involve validation of presented reasoning on human vital sign data, in order to confirm the reliability of our solution.

One of the limitations of this study is the fact, that in order to utilize the presented solution, one needs to know if either hemorrhage or endotoxin-induced sepsis is happening in the first place. This could be overcome with proper design of the fully functional predictive system, after evaluation of our approach on subjects exposed to different medical conditions contributing to changes of the arterial pressure.

Such system could compute predictions for a specific interval of time, e.g. 1 minute, and then produce a decision using weighted average. Another limitation worth mentioning is that the presented approach cannot be utilized in autopsy related analyses, as it is based on vital sign measurement.

ACKNOWLEDGMENT

The authors would like to thank the members of the Auton Lab and Robotics Institute Summer Scholar (RISS) program 2020 for all their insightful help and advice. First author would like to especially thank Xinyu Li and Dr. Artur Dubrawski for incredible amount of help, advice, and mentoring throughout the program. Furthemore, first author would like to express enormous gratitude to Rachel Burcin and Dr. John Dolan for managing the RISS program with incredible professionalism, dexterity, and care about its participants. This work was supported by Carnegie Mellon University Robotics Institute and partially supported by the U.S. Department of Defense (W81XWH19C0101), DARPA (FA8750-17-2-0130), and NIH (R01GM117622).

REFERENCES

[1] J. S. Baekgaard, T. G. Eskesen, J. M. Lee, D. D. Yeh, H. M. A. Kaafarani, P. J. Fagenholz, L. Avery, N. Saillant, D. R. King, and G. C. Velmahos, "Spontaneous retroperitoneal and rectus sheath hemorrhage-management, risk factors and outcomes," vol. 43, no. 8, pp. 1890–1897.

[2] A. B. Johnson and B. Burns, "Hemorrhage," in *StatPearls*. StatPearls Publishing. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK542273/

[3] J. W. Cannon, "Hemorrhagic shock," vol. 378, no. 4, pp. 370–379, publisher: Massachusetts Medical Society _eprint: https://doi.org/10.1056/NEJMra1705649. [Online]. Available: https://doi.org/10.1056/NEJMra1705649

[4] S. T. Cookson, J. J. Nora, J. A. Kithas, M. J. Arduino, W. W. Bond, P. H. Miller, J. Monahan, R. E. Hoffman, T. Curiel, D. Kaufman, B. M. Groves, and W. R. Jarvis, "Pyrogenic reactions in patients undergoing cardiac catheterization associated with contaminated glass medicine cups," vol. 42, no. 1, pp. 12–18, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-0304%28199709%2942%3A1%3C12%3A%3AAID-CCD5%3E3.0.CO%3B2-C. [Online]. Available: https://doi.org/10.1002/(SICI)1097-0304(199709)42:1¡12::AID-CCD5¿3.0.CO;2-C

[5] Centers for Disease Control and Prevention (U.S.), "Antibiotic resistance threats in the united states, 2019." [Online]. Available: https://stacks.cdc.gov/view/cdc/82532

[6] A. M. Abdelnoor, A. G. Johnson, A. Anderson-Imbert, and A. Nowotny, "Immunization against bacteria- and endotoxin-induced hypotension." vol. 32, no. 3, pp. 1093–1099. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC351563/

[7] A. Perner, M. Cecconi, M. Cronhjort, M. Darmon, S. M. Jakob, V. Pettilä, and I. C. C. van der Horst, "Expert statement for the management of hypovolemia in sepsis," vol. 44, no. 6, pp. 791–798. [Online]. Available: https://doi.org/10.1007/s00134-018-5177-x

[8] B. Cai, E. A. Deitch, and L. Ulloa. Novel insights for systemic inflammation in sepsis and hemorrhage. ISSN: 0962-9351 Library Catalog: www.hindawi.com Pages: e642462 Publisher: Hindawi Volume: 2010. [Online]. Available: https://www.hindawi.com/journals/mi/2010/642462/

[9] L. Vemulapalli, "LITERATURE SURVEY ON MACHINE LEARNING BASED TECHNIQUES IN MEDICAL DATAANALYSIS," pp. 1–4. [Online]. Available: https://dx.doi.org/10.36106/ijar/2113665

[10] W. Kuo, C. Häne, P. Mukherjee, J. Malik, and E. L. Yuh, "Expert-level detection of acute intracranial hemorrhage on head computed tomography using deep learning," vol. 116, no. 45, pp. 22 737–22 745, publisher: National Academy of Sciences Section: Biological Sciences. [Online]. Available: https://www.pnas.org/content/116/45/22737

[11] C. Nagpal, X. Li, M. R. Pinsky, and A. Dubrawski, "Dynamically personalized detection of hemorrhage," in *Machine Learning for Healthcare Conference*, pp. 109–123, ISSN: 1938-7228 Section: Machine Learning. [Online]. Available: http://proceedings.mlr.press/v106/nagpal19a.html

[12] J. Fagerström, M. Bång, D. Wilhelms, and M. S. Chew, "LiSep LSTM: A machine learning algorithm for early detection of septic shock," vol. 9, no. 1, p. 15132, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-019-51219-4

[13] S. M. Lauritsen, M. E. Kalør, E. L. Kongsgaard, K. M. Lauritsen, M. J. Jørgensen, J. Lange, and B. Thiesson, "Early detection of sepsis utilizing deep learning on electronic health record event sequences," vol. 104, p. 101820. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0933365719303173

[14] K. E. Henry, D. N. Hager, P. J. Pronovost, and S. Saria, "A targeted real-time early warning score (TREWScore) for septic shock," vol. 7, no. 299, pp. 299ra122–299ra122, publisher: American Association for the Advancement of Science Section: Research Article. [Online]. Available: https://stm.sciencemag.org/content/7/299/299ra122

[15] R. Gauer, "Early recognition and management of sepsis in adults: The first six hours," vol. 88, no. 1, pp. 44–53. [Online]. Available: https://www.aafp.org/afp/2013/0701/p44.html

Cardiothoracic Surgery Analysis for Predicting Acute Renal Failure Outcomes

Willa Potosnak[1], Anthony Wertz, MS[2], James K. Miller, PhD[2], Arman Kilic, MD[3], Keith A. Dufendach, MD[3] and Artur Dubrawski, PhD[2]

***Abstract*— Acute renal failure is a serious medical complication that can occur following coronary artery bypass grafting (CABG) surgery and can pose other serious medical complications if left undiagnosed and untreated. Risk models based on logistic regression were developed by the Society of Thoracic Surgeons (STS) to provide information on the potential mortality and morbidity outcomes of patients for cardiac surgeries. Previous work strove to improve the STS risk models with machine learning algorithms using pre-operative data similar to that used to develop the STS risk models. In this research, an intra-operative dataset with data obtained during CABG surgery was analyzed that is separate from that used to develop the STS risk models and previous work. The focus of this research was to determine through intra-operative data analysis whether surgical procedures and/or patient condition changes during surgery are associated with acute renal failure outcomes. Information from the analysis was used to generate a binary classification model for the purpose of assisting the pre-operative model in identifying patients' risk of developing renal failure following CABG surgery. The research identified 20 features of interest with significant (p-value < 0.05) deviations between renal failure (RF) and non-renal failure (NRF) patients during CABG surgery. The model accurately identifies approximately 10 percent of RF patients at a false positive rate (FPR) of 1 percent and approximately 22 percent at a higher FPR of 10 percent based on their surgical parameter and patient condition measurements. The model has the potential to be used as an overlay to the pre-operative model and current practices to help identify patients with higher risk of RF, thereby allowing clinicians to increase preventative care measures for these patients.**

I. INTRODUCTION

Coronary artery bypass grafting (CABG) is the most common type of heart surgery in the U.S. [1]. Approximately 340,000 procedures are performed each year [2]. Serious medical complications from CABG surgery can occur, including stroke, heart attack, acute renal failure, and death. This project focuses on acute renal failure, which is defined as a significant post-operative increase in serum creatinine or the post-operative requirement for dialysis [3, 4]. The complication of acute renal failure was chosen to analyze for this project due to a request from cardiothoracic surgeons for a predictive model that could provide additional insight for improving surgical parameters during and following CABG surgery for patients classified as likely to develop renal failure. Acute renal failure reduces the kidneys' ability to filter waste products and balance fluid and electrolytes. It also increases risks associated with other serious health complications, such as permanent kidney damage, if not diagnosed and treated immediately [5]. Because of the importance of renal function in maintaining homeostasis in the body, acute renal failure is an independent risk factor for post-operative mortality for patients requiring dialysis or other renal replacement therapies [3, 4].

Risk models based on logistic regression were developed by the Society of Thoracic Surgeons (STS) to provide information on the potential mortality and morbidity outcomes of cardiac surgery patients. The STS risk models provide a risk assessment capability based on patients' characteristics that enables doctors to better judge patients' fitness for specific types of cardiac surgeries. Previous work [3, 6] strove to improve predictions for 7 outcomes of the STS risk models, including post-operative acute renal failure and mortality, using pre-operative data similar to that used to develop the STS risk models. This previous work used the machine learning algorithm Extreme Gradient Boosting (XGBoost) to develop models which showed improved classification results for acute renal failure and modest improvement for mortality compared with results based on the existing STS models [3, 6].

The focus of this research was to determine through intra-operative data analysis whether surgical procedures and/or patient condition changes during surgery are associated with acute renal failure outcomes. Intra-operative data, which consist of data collected during CABG surgeries separate from the data used in developing the STS risk models and previous work, are used in the analysis. The intra-operative data employed in this project consist of patient demographics as well as surgical medications and surgical parameters recorded during CABG surgery for 362 patients.

This work is novel because it focused on identifying intra-operative feature differences between the renal failure (RF) and non-renal failure (NRF) patient classes for use in generating a binary classification model to predict post-operative acute renal failure outcomes. A binary classification model using solely intra-operative features has the potential to be used as an overlay to the pre-operative

[1]Willa Potosnak is a student in her 3rd year in the Biomedical Engineering Department at Duquesne University, Pittsburgh, PA, USA potosnakw@duq.edu

[2]Anthony Wertz, MS, James K. Miller, PhD, and Artur Dubrawski, PhD are with the Auton Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA awertz@cmu.edu, mille856@andrew.cmu.edu, awd@cs.cmu.edu

[3]Arman Kilic, MD and Keith A. Dufendach, MD are with the Division of Cardiac Surgery, University of Pittsburgh Medical Center, Pittsburgh, PA, USA kilica2@upmc.edu, dufendachka@upmc.edu

model and current practices to help identify potential RF patients. The intra-operative data used in this project contain time-series features (i.e., surgical/patient measurements), many of which are different from those collected prior to surgery. Information on the minute discrepancies in surgical parameters and patient condition changes between RF and NRF patients during surgery can be discerned through time-series analysis. Time-series analysis results can influence medication and procedures applied during and following CABG surgery to mitigate the risk of RF. Patient condition changes, especially those recorded just prior to, during and directly after cardiopulmonary bypass time, are of special interest for identifying the best surgical parameters.

The objectives of this research were to: 1) analyze intra-operative data from patients undergoing CABG surgery; 2) determine whether surgical procedures and patient condition changes are associated with renal failure outcomes; and 3) model determined features related to renal failure outcomes in an explainable format for better prediction and prevention of renal failure.

II. Data

The data used in this research consist of 362 patients who underwent isolated CABG surgery, meaning CABG is the only performed procedure. There is 4:1 propensity matching (4 NRF patients are present for every 1 RF patient) based on pre-operative risk scores generated from the STS risk model. Propensity matching is used to help remove potential bias that could cause a classification model to overfit the data and perform sub-optimally when applied to patients external to the project data. 75 patients developed acute renal failure following surgery and comprise the test group while the other 287 patients did not develop acute renal failure and comprise the control group. It is important to note that surgery duration varies for each patient and only half of the patients in the data have surgery durations that exceed approximately 4.5 hours as shown in Figure 4. Furthermore, CABG surgical procedures are not time-specific, but depend on the condition of the individual patient. The data subset used in the analysis consists of patient condition measurements, surgical parameters, and the top 5 medication features that affect heart rate and blood pressure. Negative time values indicate that the measurement was recorded prior to the first incision which occurs at 0 minutes and is specified in Figures 3 and 4. Patient condition measurement and surgical parameter data were forward filled for each patient by propagating the last valid measurement forward to avoid data sparsity given these features are continuous [7]. Medication features were incorporated into the dataset for the specified infusion start and stop times.

III. Analysis Methods

Patient condition, surgical parameter and medication measurements were incorporated into a dataset as individual features. An analysis of these features was conducted to discern if feature values differed between RF and NRF patients. Several methods were used for the analysis. Median value time-series plots were generated for features that showed distinct value separation for RF and NRF patients. P-values were generated for each feature using the Kolmogorov-Smirnov test on the distributions for RF and NRF patients. This tests the null hypothesis that two independent samples are drawn from the same continuous distribution. Features with significant (p-value < 0.05) deviations between classes indicate that only less than 5 percent of the time would the same distribution generate the two class samples. P-values less than 0.05 were used to help confirm features of interest and are shown in Table I. Time-series p-value plots were also generated for each feature using the Kolmogorov-Smirnov test for the two patient class distributions at each minute. These provide more minute visualizations of significant deviations between classes throughout surgery duration.

A separate analysis of medication features was conducted using the Fisher Exact test to determine if there is 1) a statistically significant association between patient class and the presence of specific medication features; and/or 2) a statistically significant association between patient class and the 20 percent highest and lowest total medication amounts administered. The p-values for these tests are shown in Table II. Medication features were not used to train the model due to their sparsity and poor effect on model performance.

6 classifiers were tested with various dataset adjustments. The classifiers tested include Logistic Regression, Random Forest, Extremely Randomized Trees (Extra Trees), Gaussian Naïve Bayes, K-Nearest Neighbors (KNN), and Quadratic Discriminant Analysis (QDA). Logistic Regression has been the model of choice for cardiac surgery risk modeling, such as the STS models [7, 8]. Even with the high calibration seen with the STS models using logistic regression [14], there are downsides to logistic regression as it requires a linear relationship between covariates and is prone to overfitting for multicollinear and large datasets like the one used in this research [9]. For these reasons, additional classifiers were tested to determine the most optimal classifier for this dataset. 10-fold cross-validation was applied to the data when training and testing the model: the dataset was split into 10 groups, the model was trained on 9 groups, and then tested on 1 group with this this process repeating a total of 10 times.

The performance of each classifier in predicting the RF class was assessed based on the true positive rate (TPR) percent values at false positive rates (FPR) of 0.01 and 0.10 percent as this indicates how well the model classifies true positives (RF patients) while still having a low FPR (i.e., the rate of classifying NRF patients as RF patients). The TPR at a low FPR can be seen clearly on an ROC Curve with the x-axis set to the $\log_{10}$ scale as shown in Figure 2 for the classifier with the best performance. The true negative rate (TNR) percent value at a false negative rate (FNR) of 0.01 percent was also taken into consideration as to how well the model classifies true negatives (NRF patients) while still having a low FNR (i.e., the rate of classifying RF patients as NRF patients). While the the main objective was to classify RF patients, a second application

of the model for helping to clear patients from consideration of developing RF would also be useful for this objective. Various dataset adjustments were tested with these classifiers. The main dataset adjustments that were used to determine the best model are:

1) Computing a rolling standard deviation for each of the original numeric attributes to generate new features for model training, referred to as featurization. This serves to generate features that indicate patient condition measurements that deviate from the mean feature values. This technique is used as classifiers may discern class differences better for certain features in this format.

2) Using scikit-learn [10] Robust Scaler with pre-operative data, or data collected in the 10-minute window prior to each patient's surgery start time. Robust Scaler removes the median and scales the data to the quantile range making it robust to outliers present in the dataset. The values from the 10-minute windows serve as patient baseline values that when applied to scale the dataset, result in features that indicate patient conditions that deviate from baseline values during surgery.

3) Applying the scikit-learn [10] Recursive Feature Elimination (RFE) tool for specific classifiers and using only the RFE-selected features when training and testing the model. When applied to the training set of data, RFE eliminates features that are least important to the model and returns features with the highest importance for the specified estimator (classifier). This tool can help improve model performance by reducing noise in the dataset due to sparse and/or irrelevant features.

Additional adjustments include training the model at certain time intervals that showed improved class separation. Time intervals with improved class separation were determined through an analysis of the median predicted positive probabilities of the patients: predicted positive probabilities for RF patients close to 1 and a predicted positive probabilities for NRF patients close to 0 is ideal. The median positive probability plot which assessed all patients' probabilities of developing renal failure throughout the surgery duration for the classifier with the best performance is shown in Figure 3.

IV. Results

Features with significant (p-value < 0.05) deviations between classes, meaning that only less than 5 percent of the time would the same distribution generate the two class samples, are listed in Table I with their respective p-values and difference between mean values. Classifiers trained and tested on only features with p-values considered significant showed poor classification performance and a low percentage of correctly identified RF patients.

A specific analysis of medication features using p-values generated from the Fisher Exact Test showed that the presence of medication features for either class is not statistically significant, nor is the number of RF patients in the groups of patients administered the 20 percent highest and lowest total dosages/volumes. The medication features and their p-values are shown in Table II. P-value 1 indicates statistical significance between patient class and the presence of specific medication features. P-values 2 and 3 indicate statistical significance between patient class and the 20 percent highest and lowest total medication amounts administered, respectfully.

The classifier that showed the best model performance is Extra Trees with additional standard deviation features for each feature in the dataset, no scaling of the dataset with pre-operative data and the dataset containing only RFE-selected features. This model is shown in Figure 1. The top 30 RFE-selected features and their feature importances as determined by Extra Trees are shown in Figure 5. Extra Trees shows the best model performance in terms of having the highest TPR at an FPR of 0.01 percent out of all classifiers tested for this combination of dataset adjustments. This classifier accurately identifies approximately 10 percent of RF patients at an FPR of 1 percent and approximately 22 percent at a higher FPR of 10 percent as shown in the first plot in Figure 1, which is enlarged in Figure 2. In addition, it accurately identifies approximately 10 percent of NRF patients at an FNR of 1 percent as shown in the third plot of Figure 1. The approximation in identification is due to the randomized nature of Extra Trees in terms of selected features and cut-point choice that is explained further in [11]. Class separation based on the probability estimates for the positive (RF) class as determined throughout surgery duration by the Extra Trees classifier is shown in Figure 3. Distinct class separation, especially in the 3-7 hour time interval, indicates that the classifier can discern a distinction between patients, and that other machine learning methods may improve model performance.

Significant features were used to assess credibility of the Extra Trees model in classifying patients based on its ranked feature importances as shown in Figure 5. Of the top 30 features with the largest importances out of all 54 RFE-selected features, 9 features with significant deviations between classes were present.

V. Discussion

The model has the potential to be used as an overlay to the pre-operative model and current practices to help identify patients with higher risk of RF, thereby allowing clinicians to increase preventative care measures for these patients. In addition, this classifier can help identify NRF patients, which can allow clinicians to better allocate preventive measures to patients who show higher risk of RF as well as those not identified by the classifier. Moreover, the model performance indicates that an entire intra-operative time series analysis may not be the best approach, and that the surgery time-series analysis should be partitioned into sections based on surgical procedures. Because there is a large variation in patient surgery durations as shown in Figure 4, there could be distinct variations in patient conditions at their respective stages. Including the entire intra-operative time series in the analysis could be convoluting the data. So,

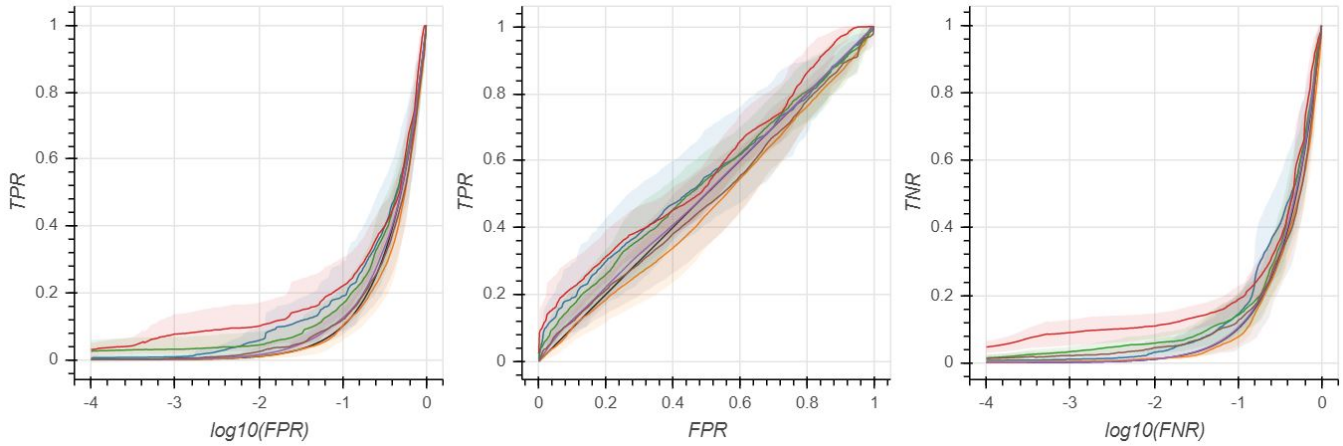


Fig. 1: ROC curves for all 6 classifiers. Shaded region is the standard error.

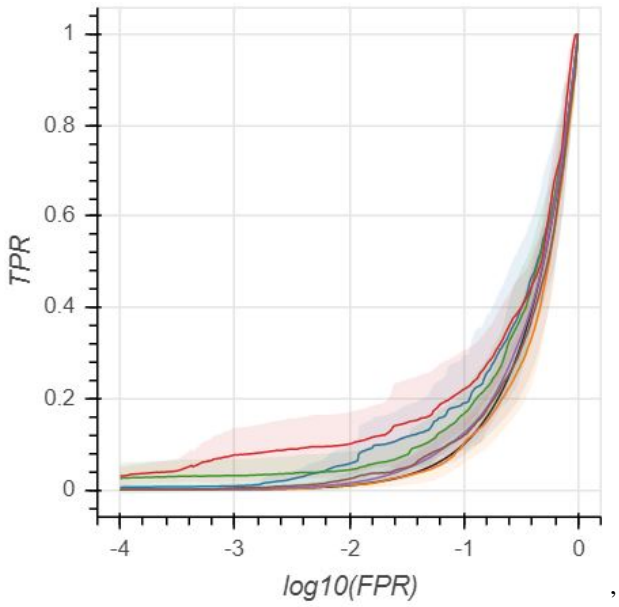
Feature	P-Value	Mean Difference
Stroke Volume	-7.26	0.62
RV End Diastolic Volume	-6.88	2.04
Pulse Pressure-Blood	-5.49	2.14
Central Venous Pressure	-4.54	0.76
NIRS Cerebral Oxygenation-L	-3.96	2.37
Mean Blood Pressure	-3.96	0.76
SvO2	-3.69	1.12
Systolic Blood Pressure	-3.69	1.45
Oxygen Percent (FiO2)	-3.69	1.43
RV Ejection Fraction	-3.42	0.50
Arterial Diastolic Pressure	-3.17	0.70
BIS Value	-3.17	1.28
Heart Rate-Pleth	-2.92	3.41
Diastolic Blood Pressure	-2.92	0.68
NIRS Cerebral Oxygenation-R	-2.46	1.90
Heart Rate	-2.24	1.78
Mean Arterial Pressure	-2.24	0.55
Pulse Pressure-Arterial	-2.03	1.10
Pulmonary Artery Mean	-1.83	0.23
Epinephrine 64 Dose	-1.64	3.13

TABLE I: $\log_{10}$(p-values) and mean differences between RF and NRF patients for significant features

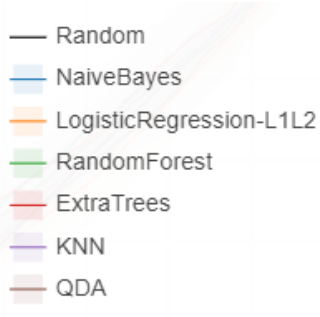
Feature	P-value 1	P-value 2	P-value 3
Phenylephrine Volume	1.00	0.712	0.856
Phenyleprhine Dosage	1.00	0.461	0.856
Vasopressin Volume	0.007	0.545	0.360
Vasopressin Dosage	0.007	0.545	0.360
Epinephrine 10 mcg/mL Volume	0.018	0.526	1.00
Epinephrine 10 mcg/mL Dosage	0.018	0.526	1.00
Epinephrine 64 mcg/mL Volume	0.019	0.377	0.517
Epinephrine 64 mcg/mL Dosage	0.027	0.828	1.00
Norepinephrine Volume	0.005	0.450	0.450
Norepinephrine Dose	0.005	0.205	0.405
Albumin 5 percent Volume	0.007	1.00	0.819

TABLE II: P-values for medication features. None are considered statistically significant after Bonferroni correction (p-value = 0.0045).

while the model correctly identifies approximately 10 percent of RF patients at a FPR of 1 percent and approximately 22 percent at a higher FPR of 10 percent, partitioning the data based on procedures within the surgery could improve



(a) Plot 1 of Figure 1 on a larger scale. Approximately 10 percent accurate identification of RF patients at a FPR of 0.01 percent and 22 percent at an FPR of 0.10 percent for Extra Trees Classifier



(b) ROC curve legend

Fig. 2: Extra Trees classifier has the best ROC curve results out of the 6 tested classifiers

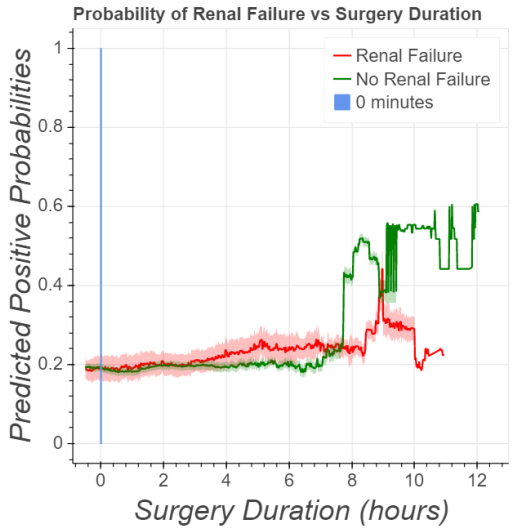


Fig. 3: Positive probability estimates for the optimal model with Extra Trees classifier. 0 minutes indicates first incision of surgery.

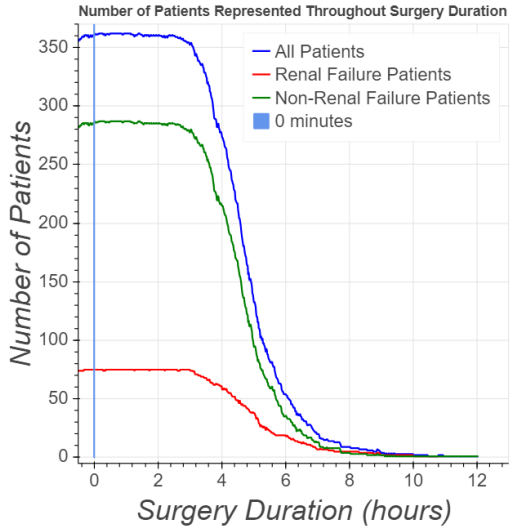


Fig. 4: The number of patients present throughout surgery. 0 minutes indicates first incision of surgery.

model performance. Partitioning may also provide results that clinicians can more easily use to discern what and when surgical parameters should be adjusted.

The model performance could also indicate that features currently collected during CABG surgery may not be representative of renal function. Collecting and evaluating features more directly correlated to renal function during CABG surgery could improve the intra-operative analysis and, ultimately, the model performance. For example, serum creatinine level directly indicates glomerular filtration rate, which directly correlates to renal function [9, 13, 15]. Glomerular filtration rate decreases as renal function decreases, leading to an increased serum creatinine level, which is a strong risk factor for RF [16]. Creatinine level is

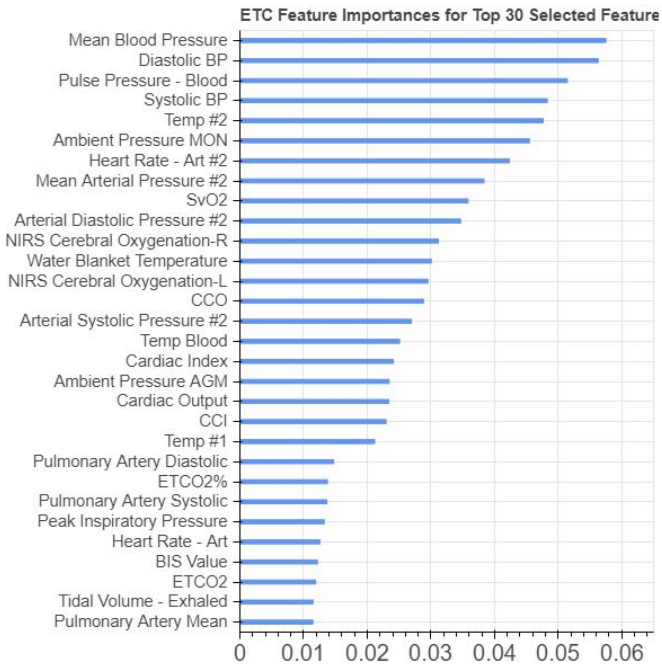


Fig. 5: Extra Trees RFE selected features ranked by importance determined by the classifier with respect to all features

measured pre-operatively and used in pre-operative models to predict RF outcomes [3, 9, 12, 14, 15]; it is also measured post-operatively to monitor RF, so its inclusion in time-series intra-operative data could potentially improve model performance.

VI. Future Work

Future work will involve partitioning the data for each patient into 5 stages based on CABG surgical procedures. Recorded features and their values will indicate the start and end points of the 5 stages for each patient. An analysis of patient condition, surgical parameter and medication features will be performed for each stage to better assess patient changes regarding specific events during CABG surgery. Classifiers will be trained on each of the surgical stages to refine the model and minimize possible data convolution. The XGBoost algorithm will also be applied as it has improved model performance as shown for work in [3, 6, 9].

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. Thank you to Carnegie Mellon Robotics Institute for funding this research. A special thanks to Rachel Burcin and Dr. John Dolan for their efforts to help make the RISS program possible this summer via a virtual approach.

References

[1] "Bypass Surgery Shows Advantage," *nih.gov*, 2012.
[2] "New Study Shows Approximately 340,000 CABG Procedures per Year in the United States," *idataresearch.com*, 2018.

[3] A. Kilic, A. Goyal, J. K. Miller, T. G. Gleason, and A. Dubrawski, "Performance of a Machine Learning Algorithm in Predicting Outcomes of Aortic Valve Replacement," *The Annals of Thoracic Surgery*, 2020.

[4] Duca, S. Iqbal, E. Rahme, P. Goldberg, and B. Varennes, "Renal Failure After Cardiac Surgery: Timing of Cardiac Catheterization and Other Perioperative Risk Factors," *The Annals of Thoracic Surgery*, vol. 84, no. 4, p. 1264-1271, 2007.

[5] "Acute kidney failure," *mayoclinic.org*.

[6] A. Kilic, A. Goyal, J. K. Miller, E. Gjekmarkaj, W. Lam Tam, T. G. Gleason, I. Sultan, and A. Dubrawski, "Predictive Utility of a Machine Learning Algorithm in Estimating Mortality Risk in Cardiac Surgery," *The Annals of Thoracic Surgery*, vol. 109, no. 6, p. 1811-1819, 2020.

[7] D. M. Shahian, E. H. Blackstone, F. H. Edwards, F. L. Grover, G. L. Grunkemeier, D. C. Naftel, S. A.M. Nashef, W. C. Nugent, and E. D. Peterson, "Cardiac Surgery Risk Models: A Position Article," *The Annals of Thoracic Surgery*, vol. 78. no. 5, p. 1868-1877, 2004.

[8] D. M. Shahian, J. P. Jacobs, V. Badhwar, L. Feng, X. He, S. M. O'Brien, et al., "The Society of Thoracic Surgeons 2018 Adult Cardiac Surgery Risk Models: Part 1-Background, Design Considerations, and Model Development," *The Annals of Thoracic Surgery*, vol. 105, no. 5, p. 1411-1418, 2018.

[9] Lee, H. Yoon, K. Nam, Y. J. Cho, T. K. Kim, W. H. Kim, and J. Bahk, "Derivation and Validation of Machine Learning Approaches to Predict Acute Kidney Injury after Cardiac Sugery," *Journal of Clinical Medicine*, vol. 7, no. 322, 2018.

[10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] P.Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach Learn*, vol. 63, p. 3-42, 2006.

[12] C.V. Thakar, S. Arrigain, S. Worley, J. Yared, and E. Paganini, "A clinical score to predict acute renal failure after cardiac surgery," *Journal of American Society of Nephrology: JASN*, vol. 16, no. 1, 2005.

[13] T. Coulson, M. Bailey, D. Pilcher, C. M. Redi, S. Seevanayagam, J. Williams-Spence, and R. Bellomo, "Predicting Acute Kidney Injury After Cardiac Surgery Using a Simpler Model," *Journal of Cardiothoracic and Vascular Anesthesia*, 2020.

[14] S. M. O'Brien, L. Feng, X. He, N. D. Desai, F. H. Edwards, D. M. Shahian, et al., "The Society of Thoracic Surgeons 2018 Adult Cardiac Surgery Risk Models: Part 2-Statistical Methods and Results," *The Annals of Thoracic Surgery*, vol. 105, no. 5, p.1419-1428, 2018.

[15] J. Chikwe, J.G. Castillo, P.B. Rahmanian, A. Akujuo, D. H. Adams, and F. Filsoufi, "The impact of moderate-to-end-stage renal failure on outcomes after coronary artery bypass graft surgery," *Journal of Cardiothoracic and Vascular Anesthesia*, vol. 24, no. 4, p. 574-579, 2010.

[16] R. Bellomo, C. Ronco, J. A. Kellum, R. L. Mehta, and P. Palevsky, "Acute renal failure – definition, outcome measures, animal models, fluid therapy and information technology needs: the Second International Consensus Conference of the Acute Dialysis Quality Initiative (ADQI) Group," *Critical Care*, vol. 8, no. 4, 2004.

Adaptive Agent Architectures for Real-time Human-Agent Teaming

Suhas Raja[1], Siddarth Agarwal[2], Tianwei Ni[2], Yikang Gui[2], Huao Li[2], Fan Jia[2], Dana Hughes[2], Katia Sycara[2]

***Abstract*— Various approaches have been established to create game-playing agents. However, many of these fail to recognize changepoints in teammate behavior or environments, leading to suboptimal performance in games with these mechanics. We propose two distinct similarity mechanisms that an intelligent agent may use to estimate the policy of others in a real-time shared game, given a library of known policies. Each of these mechanisms are wrapped into adaptive agents that are then tested with anonymous human players. These agents perform notably better than individual agents from the library, indicating that they adapt to the human mid-game.**

After establishing and analyzing each of these methods, the authors investigate a new framework for Team Space Fortress that leverages these mechanisms to provide a novel layer of adaptivity. By introducing a version of Monte-Carlo Tree Search, with substantial modifications, the authors use the similarity mechanisms to accelerate the search for Subgame Perfect Equilibria in a real-time strategy game. The framework offers significant promise to create cognitive agents that reason according to not only the behavior of their teammates, but also to implicitly negotiate outcomes with their opponents.

I. INTRODUCTION

Multiagent systems are crucial to deploying scalable intelligent robotic systems. However, most real-world environments are not exclusively composed of known agents. For example, unmanned aerial vehicle teams may maintain semi-autonomy, but must be capable of adapting to novel changes in their environment. These changes may span novel environmental stimuli to unpredictable behavior of human agents, whether teammates or opponents. Adaptive capabilities are essential to safely deploy and collaborate in a real-time environment. The fundamental challenge for a robot to work with a human, instead of simply another robot, is that humans may have more complex and variable behaviors or intent. To succeed, cooperative agents must be able to estimate and predict human behavior to inform their action accordingly.

Past research on human-agent teaming generally focuses on processing retrospective teammate reports, where software analyzes historical observations of humans to inform behavior in the present. [1]–[3] These historical behaviors may fail to capture potential changepoints in an environment and limit the ability of software to truly adapt to the situation. Real-time adaptation is critical in practical deployments, particularly those for military application. In particular, the ability to respond to real-time observations improves an agent's ability to perform in the face of novel team structures or situations. The fundamental goal of this research is to define and analyze the performance of a novel adaptive agent architecture that performs well in a nontrivial real-time strategic game, Team Space Fortress (TSF).

In this paper, we investigate behavior cloning and clustering approaches that estimate a human's policy in real time. We then provide an adaptive finite extensive-form formulation of the Team Space Fortress Strategy Game that facilitates the usage of this information to adaptively work with a teammate. This extends our past work [4]–[6] to a adaptive adversarial environment, requiring that prototyped agents are fully responsive to rapidly changing states, without explicit communication.

Our adaptive agent leverages a policy library of rule-based and reinforcement learning (RL) agents that can perform reasonably in TSF when paired with a human. By observing human behavior in real-time, the agent may estimate similarity indices between the human and each agent in the policy library. In this policy library, there are 8 shooter agents and 9 bait agents, each of which may be built with rule-based logic or pretrained reinforcement learning models. The adaptive agent must leverage these indices to improve performance beyond that of each individual agent selected from the library. The naive adaptive agent simply acts according to whichever agent historically performed best against the teammates estimated type. The Extensive-form Monte Carlo Tree Search (FEFMCT) formulation defines an extended version of TSF where both the fortress and software agent leverage Monte Carlo-esque rollouts to search for equilibria within a finite extensive-form game, given the agent's predictions of how the teammate will perform.

After formulating the naive and FEFMCT adaptive agents, they will be tested by having human players play against different agents online. These players are sourced through Amazon's Mechanical Turk program (a labor contracting site) and play TSF through their internet browser. Players will not be told which agents they are playing with and will be rotated through different sequences of players to ensure agent anonymity. By developing adaptive agents and analyzing them through this framework, this paper aims to demonstrate that our agent architecture succeeds in facilitating real-time adaptive ability in software agents.

The primary contribution of this paper is to define, implement, and analyze the efficacy of two novel similarity mechanisms that can provide adaptive capability to autonomous agents in a game. The secondary contribution is the definition and analysis of the FEFMCT adaptive system, and the

[1]Suhas Raja is with the Department of Electrical Engineering at The University of Texas at Austin. sraja@utexas.edu

[2]Dr. Katia Sycara, Tianwei Ni, Siddharth Agarwal, and Dr. Dana Hughes are with the Robotics Institute at Carnegie Mellon University.

introduction of a new edition of TSF.

II. Background

The following section discusses background concepts that are necessary to understanding the approaches described in sections (3) and (4).

A. Team Space Fortress (TSF)

The player and autonomous agent each control a ship that moves around a low friction 2d environment. At the center of the stage lies a rotating fortress. The fortress and ships can all fire missiles. However, when ships fire, they incur a score penalty of 10. Similarly, when a fortress fires it exposes a vulnerable area on its backside, where ships may fire a missile to earn a kill worth 110 points. Alternatively, ships can die and lose 100 points by either being hit by a missile or running into the fortress. Each game lasts three minutes, with the ships goal to maximize their score. A sample screen from the game is shown in figure 1.

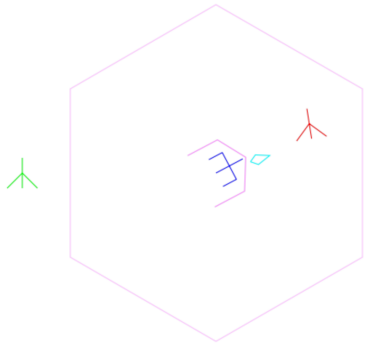


Fig. 1. TSF Sample Game Screen

Once the fortress has been eliminated, both players must leave the outer green boundaries before it respawns. The default fortress policy is to target the first agent that entered the boundaries, among those that are within the boundaries.

B. Monte-Carlo Tree Search

Monte-Carlo Tree Search is a widely used algorithm to iteratively explore a tree of game states. The root node of the tree denotes the current state. At each iteration, the algorithm will select a node according to an external metric and expand it by creating nodes for each potential following state. Many selection metrics have been implemented and profiled to examine their performance in MCTS. [7] However, in this paper we use one of the most commonly used selection metrics, the Upper-Confidence Bound 1 (UCB1) metric, which is defined by the formula below. w_i indicates the sum of scores across all s_i trials at or below node i, with similar variables for the parent p.

$$\frac{w_i}{s_i} + c * \sqrt{\frac{\ln s_p}{s_i}}$$

Note that there exists a parameter in the latter term, c, which adjusts the algorithm's priorities between risk and reward— whether it should prefer to expand relatively well-known states, or states that may have more variance in the value estimate.

C. Extensive Form Subgame Perfect Equilibria

A Subgame Perfect Equilibrium (SPE) extends the concept of a Nash Equilibrium to Extensive Form Games, that is, games that have multiple steps. Extensive Form Games may either have finite or infinite stages. The formulation in section (3) relies on a Finite Extensive Form conceptualization.

Consider a two-agent game that takes N steps. Each player has a predefined policy indicating how they ought to behave at each decision node in the game tree. A subgame perfect equilibrium requires that, at any of these decision nodes, neither player could improve their outcome by deviating from their policy, given that the other player maintains their policy at all nodes. An example of an SPE in a small payoff tree is shown in figure 2.

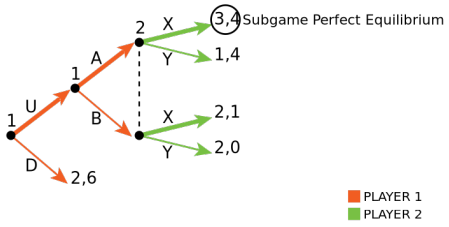


Fig. 2. Example Payoff Tree with Subgame Perfect Equilibria Marked [8]

Player one takes actions U and A, and player two takes action X. The dashed lines indicate that player two cannot tell which subtree they are in. Given these policies and payoffs, it remains true that neither agent would want to select an alternate action at any node. Player two could only be worse off by selecting Y. Given player two's behavior, no other player one strategy can beat the payoff of three earned by the actions (U, A).

D. Autoencoders

An autoencoder consists of two complementary neural networks, an encoder and decoder. [9] These networks are trained simultaneously, where the source and target data are identical. An encoder transforms the input into a lower dimensional structure, and the decoder interpolates the original data from that structure. By training the output of the decoder based on the input to the encoder, as pictured below, the networks capture the semantics in a low-dimensional space, allowing generalized analysis. In TSF, the input and output data is comprised of the players positions, statuses, and actions.

III. Methods

The following subsections establish the design proposed and implemented for this research. Two similarity mechanisms used in the basic adaptive agents are defined. Finally, we define a game architecture that leads the autonomous players to seek an equilibrium, given the policy the human is estimated to play according to similarity metrics.

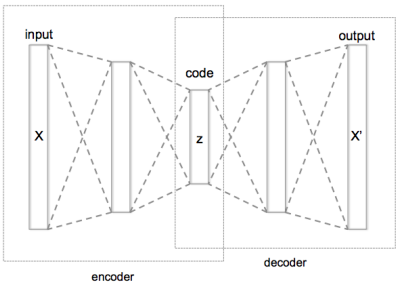


Fig. 3. Autoencoder Layout [10]

A. Similarity Metrics

The first similarity metric is based on behavior cloning (BC). BC applies the following equation to measure the distance between observed human behavior, H, and an agent policy, B.

$$dist(H, B) = E_{s,a\ H,B}[logP_B(a|s)]$$

The implementation of the approach functions as follows. Each policy, B, provides an probability distribution across actions given a particular game state. A collection of human observations is stored, that contain the game state (s) and the associated human action (a). For a fixed policy, B, the average log probability that the policy would have matched the human behavior is calculated across all observed states. The resulting metric is considered as the policy distance between an agent and human. The distance metric is then used to identify the agent with the greatest average consensus to the human (as described above), information which can be used by an adaptive agent. The second similarity metric is the Autoencoder Similarity Metric (ASM). ASM uses an autoencoder trained to identify an agent given the historically observed behavior from known agents. By translating observed behavior into the latent space of the autoencoder, the encoder effectively generates an embedding from these observations. ASM then uses this same encoder on the observed human behavior from a set range of recent frames and selects the agent with the lowest L2 regularized distance between embeddings.

B. FEFMCT

To handle the multiagent nature of TSF, the FEFMCT edition of MCTS has notable structural adjustments. The most novel adjustment is that both the fortress and agent share a common game tree. To accomplish this, three modifications are made:

- At any state, outward transitions include the product of each pair of actions players could execute, rather than an individual player's action set.
- Expansions alternate between those that maximize and minimize the UCB1 metric.
- Each expansion and rollout fixes human player's policy according to the chosen similarity mechanism.

The first modification transforms the game tree into a payoff tree. Once the game tree is expanded to a reasonable depth, established equilibrium search algorithms can be applied to search for SPE.

The second adjustment alternates expansions between the maximum and minimum UCB1 metric to ensure that a relatively balanced and fair tree is generated. Such alternation ensures that neither agent is able to expand only biased subtrees.

The third modification ensures adaptivity. By considering states with respect to the expected behavior by the human teammate, the agent can conduct a strategy that maximizes their score based on their observations. Since the human informs the states in the tree, decisions by the agent then account for the expected human behavior.

However, before an SPE can be found, the payoff matrix must be adjusted. In a traditional payoff matrix, only leaf nodes contain payoffs. In the FEFMCT tree, each decision node includes payoffs. We introduce a parameter, λ, which allows the user to define a pruning depth. The pruning depth causes the algorithm to remove all nodes beneath that depth (since they remain relatively unexplored to shallower nodes), then uses the new leaf nodes to construct an estimated payoff tree.

If an SPE is found within this game tree, both players will follow it. This is because no profitable deviation exists by definition. This means that they will follow the policies that comprise the SPE.

To handle the case where an SPE does not exist, we introduce the concept of an approximate subgame perfect equilibrium (ASPE). In an ASPE, agents find the deepest route in the payoff tree that has no profitable deviations. Given that the tree is already pruned, with values approximated through Monte-Carlo simulation, approximating an SPE does not substantially compromise the SPE's integrity.

The FEFMCT approach leverages the following algorithm to identify an ASPE.

Algorithm 1: ASPE Algorithm

```
Result: An equilibrium path through the game state
        tree.
for each leaf L do
    L.value = L.score
end
updatesPossible = True
while updatesPossible do
    updatesPossible = False for each node N do
        if child.value exists for all children of N then
            Identify 1-step Nash Equilibrium among
             children.
            if equilibrium exists then
                updatesPossible=True
                Attach equilibria score to N.value.
            end
            else
                Prune Children. N.value = N.score
            end
        end
    end
end
Apply Depth-First Search to find the deepest path
 that meets ASPE criteria.
```

A critical challenge that arises when applying tree-search to Team Space Fortress is that it rapidly becomes computationally intractable. The game runs at 30 frames per second (fps) which means that, at a minimum, 30 expansions are necessary to forecast moves one second in the future. The agent simply cannot react fast enough since players score sporadically, up to 10-20 seconds apart, and the branch factor is inflated due to the inclusion of both players actions along any transition.

A final pair of structural adjustments substantially alleviates this challenge. While the game runs at 30fps, the tree groups frames together in sets of 10, to scale the tree depth further by a factor of 10. Given that these agents are performing in an adaptive environment, it is also helpful to have the agent operate at a lower frame rate (3fps). This ensures that the agents are less volatile, and the human player can better adapt to their teammate.

IV. Results

The following subsections discuss the data and results associated with the approaches defined in section 3.

A. Similarity Metric Performance

Each similarity metric was packaged into an adaptive agent, which was tested on Amazon Mechanical Turk. These agents measured similarity between the human behavior and each of 9 policies from the policy library to identify the approximate policy that the human follows. These similarity mechanisms require a parameter that adjusts the number of recent frames used when calculating similarity, which is tested at multiple different values. It then references a historical performance table to see which agent best complements the assumed human policy and acts according to the complementary agent.

From this experiment, we would like to test the following hypotheses:

- Our proposed adaptive agents outperform any single static policy in the policy library, when paired with human players.
- The window size parameter, which controls the agent's learning rate, has a significant influence on adaptive agents' performance.

We employed a mixed experiment design where the between-subject variable is the adaptive agent type, and the within-subject variable is the window size parameter of adaptive agent. We tested three different adaptive agents in this study: BCAgent shooter, BCAgent bait, and ASMAgent shooter. Participants were divided into three groups which corresponds to three agents. Within each group, participants will team with five variants of the designated agent in a random sequence. The five variants include three values of the window size parameter for adaptive agent (150, 400, 800ms) and two best-performed static agent policy (representing the extreme condition of 0 window size where the adaptive agent becomes static). The team performance is measured by number of fortress kills in each trial.

10 participants were recruited from Amazon Mechanical Turk. They were randomly assigned into three groups and paired with different agents. Participants were paid USD 2 for participating in the online study.

Participants were randomly assigned a role of either shooter or bait and then teamed with artificial agents in the corresponding role to play Team Space Fortress. Each participant would need to complete five sessions of data collection with three 1-min game trials in each session. Participants teamed with different agent variants between sessions, and played with each agent for three trials. The approximate length of each experiment is 15 minutes.

The performance of the shooter and bait BC agents is shown in figure 4, compared to the top two baits and top two shooters.

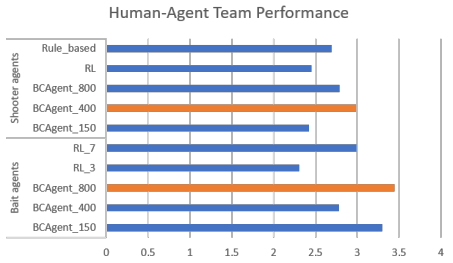


Fig. 4. Performance data for best performing static agents and BC Agent

It is clear that the bait BC agent provides a notable increase in performance over the RL_3 and RL_7 agents, by approximately 20 percent. In particular, the BC bait agent with the highest window size substantially outperformed all other baits.

The shooter BC agent generally provides a mild improvement against the other shooter agents. A lower degree of

complexity and variance in possible shooter strategies may account for the more mild increase, since the degree of adaptation may be more limited.

B. FEFMCT Analysis

The FEFMCT approach cannot be benchmarked using the same method since it relies on a different formulation of TSF. The edition of TSF used for FEFMCT allows both the agent and fortress to act according to variable strategies, in contrast to the original version of TSF where the fortress follows a static policy. Specifically, the agent and fortress each follow the actions prescribed by the FEFMCT equilibrium – that is, the fortress follows a more complex strategy for FEFMCT. The value of FEFMCT is that it introduces this more challenging version of TSF and provides a clear cognitive agent framework to adapt to both predictions of human behavior and negotiated outcomes with opponents.

V. Future Work

In the future, we intend on accelerating the FEFMCT model to operate in a real-time environment. By enabling it to run in real-time, the model can be used to command the autonomous agents while a human agent is playing. In contrast, we are currently using a synthetic agent that behaves similarly to a human.

Once the model is deployed with real humans, we intend to gather data that captures the running status of equilibria as they are calculated during the game.

ACKNOWLEDGMENT

S. Raja thanks Dr. John Dolan and Ms. Rachel Burcin of Carnegie Mellon University for supporting the Robotics Institute Summer (RISS) program that facilitated this research. S. Raja also thanks the National Science Foundation for their generous Research Experience for Undegraduates (REU) funding that made this research possible.

References

[1] S. W. Kozlowski and G. T. Chao, "Unpacking team process dynamics and emergent phenomena: Challenges, conceptual advances, and innovative methods." *American Psychologist*, vol. 73, no. 4, p. 576, 2018.

[2] S. W. Kozlowski, J. A. Grand, S. K. Baard, and M. Pearce, "Teams, teamwork, and team effectiveness: Implications for human systems integration." 2015.

[3] S. W. Kozlowski and K. J. Klein, "A multilevel approach to theory and research in organizations: Contextual, temporal, and emergent processes." 2000.

[4] S. S. Gupta, D. Hughes, and K. Sycara, "Imitation learning for latent factors in collaborative multi-agent systems."

[5] K. Sycara and M. Lewis, "Integrating intelligent agents into human teams," 2003.

[6] G. Sukthankar, R. Shumaker, M. Lewis, E. Salas, S. Fiore, and M. Letsky, "Intelligent agents as teammates," *Theories of Team Cognition: Cross-Disciplinary Perspectives*, pp. 313–343, 2012.

[7] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[8] W. Commons. (2017) Backwards induction example 2. File: Backwards Induction Example 2.png. [Online]. Available: https://en.wikipedia.org/wiki/Subgame-perfect-equilibrium/media/File:Backwards-Induction-Example-2.png

[9] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[10] B. Boehmke and B. M. Greenwell, *Hands-on machine learning with R*. CRC Press, 2019.

Trajectory Optimization for the Legged System

Jiming Ren[1], Shuo Yang[2], Zhaoyuan Gu[2], and Howie Choset[2]

***Abstract*— This paper presents two optimization formulations for legged systems using Direct Collocation method and iterative Linear Quadratic Regulator (iLQR) method respectively. The Direct Collocation-based approach generates a whole-body motion plan in a hierarchical two-phase manner, namely optimal stance and optimal transition, with the input of state estimation and global elevation mapping yet without prior knowledge on footholds and step timings. Alternatively, a discrete-time, constrained iLQR algorithm can work out an executable control policy besides a nominal optimized trajectory, however, a gait pattern should be pre-specified. In this work, we integrate the optimized trajectories given by Direct Collocation approach with the robot's slow-level controller to enable the robot to walk automatically on the unstructured terrains. We demonstrate our formulation and framework running online on a hexapod robot, Mat6, both in simulation and hardware. Our work is one of the very few examples where Trajectory Optimization (TO) is applied on hardware to plan dynamic whole-body motions and maybe one of the best performances in climbing skills for legged systems.**

I. Introduction

Legged robots as versatile systems, are primarily targeted to be deployed on unstructured terrains where they outperform tracked or wheeled robots. A core challenge in this setting is a robot to automatically reach a specified high-level goal on the terrain map without collision with the environment. This involves the generation of feasible body- and end effector-motions along with contact forces that retain the balance. TO formulation leveraging whole-body dynamics turns out to be an ideal solution as it can determine holistic motions and forces in a general way given the goal state, gait pattern, and terrain information. This work presents two approaches to solve the numerical TO problems for legged systems, namely Nonlinear Programming (NLP) and Dynamic Programming (DP). The first part of the work deals specifically with treating complex motion planning as an NLP problem and using Direct Collocation to compute the optimal trajectory. Furthermore, we realize the trajectory by a low-level tracking controller to enable a hexapod robot Mat6 (shown in Fig. 1) to walk on rocky lands, slopes, and stairs.

NLP has been widely applied for complex systems with a myriad of decision variables, and previous works on higher-dimensional legged systems include [1]–[3]. In essence, NLP constructs a cost function and searches for a set of decision variables to minimize the cost function until convergence.

[1] J. Ren is with the Department of Mechanical and Aerospace Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China. jrenaf@ust.hk

[2] S. Yang, Z. Gu and H. Choset are with the Biorobotics Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. {shuoyang, zhaoyuan, choset}@andrew.cmu.edu

Fig. 1. Mat6, a hexapod testbed consisting of 18 joints actuated by series-elastic HEBI X Modules. Each module provides measurements including angular positions, angular velocities, and joint torques. Additionally, an Intel Realsense D435i camera is mounted on top of the robot.

There are several methods to solve NLP and they mainly fall into two categories: sequential methods and simultaneous methods. Sequential methods only use the control input as a decision variable while simultaneous methods optimize over both state and input, which are coupled together via the system dynamics. Though sequential methods like Direct Shooting and its extension have been used in whole-body control of human-like robots [4], simultaneous methods are more suitable for torque control legged robots with complicated control inputs and path constraints. The most prominent approach of simultaneous methods is Direct Collocation. Direct Collocation transcribes the trajectory optimization problem from an infinite-dimension continuous optimal control problem to a finitely parameterized NLP by discretizing the state and control trajectories into a finite number of nodes and then interpolating between them using polynomial spline approximation. This approach has been widely applied to the whole-body planning for legged robot motion control involving both body- and end effector-motions [5], [6]. However, high dimensional optimization problems bring about the problem of expensive computational cost, and consequently, slow convergence rate. Besides, the outcome of the Direct Collocation approach requires the low-level controller to track the trajectory and transform it into joints torque to be leveraged by the actuators.

Another approach of TO named DP can figure out a control input in addition to an optimal nominal trajectory. While directly solving a feedback policy through the

Hamilton–Jacobi–Bellman equation could be intractable for a high dimensional system, a variant of dynamic programming called Differential Dynamics Programming (DDP) can overcome this problem by addressing the solvable Riccati equation. DDP starts with an initial guess on the control sequence, proceeding by iteratively performing a backward pass on the nominal trajectory to generate a new control policy, and then updating a new trajectory in a forward pass for the next cycle. In later work, DDP-based approaches named iLQR and Sequential Linear Quadratic Control (SLQ) are introduced to operate on a linear approximation of the state along the trajectory rather than quadratic approximation DDP operates on. The much faster computational speed of DDP compared to NLP gives the possibility of being more compatible with Model Predictive Control (MPC) settings. The latter with separation of the low-level motion tracking controller block from the trajectory generation block is unable to react proactively to the deviation of proprioceptive state estimation to the real state during the trajectory execution process. [7] has shown that SLQ in an MPC setting frequently updating the intermediate trajectory can offset this deviation.

However, the original DDP-style algorithm can not handle high-level tasks for legged systems as unconstrained controllers are limited in real scenarios. In the second part of the paper, we present the discrete-time, control-constrained iLQR algorithm which optimizes the control input given the switching pattern. To handle the contact constraints, one solution is taking into account the effects of control constraints to the cost-to-go function. [8] includes contact forces as control input variables and gradually enforces contact constraints via the cost function. Alternatively, this work adopts the discrete-time, constrained LQR idea initiated in [9] and reformulates into iLQR algorithm. Similarly, [3] has derived a continuous-time SLQ variant which allows for including state-input constraints as hard limits. However in this work, we loosen the constraints to control-only contact constraints to increase the efficiency whilst collision avoiding is considered separately for the swing legs.

The rest of this paper is organized as follows. Section II presents the transformation of the optimal whole-body motion planning from its specialized form into an NLP problem and solved by an NLP solver. Section III calculates an optimal trajectory from initial to target state using iLQR methods. Section IV demonstrates the experimental works on Mat6, which is realized in a hierarchical control architecture based on the results from the NLP solver. The conclusion is in Section V.

II. Direct Collocation Approach

NLP formulation in this section consists of two-layer nonlinear optimization programs which are periodically implemented to find the optimal motion plan. After a goal is specified in a higher level path planner, the first program finds the optimal stance posture at the position where the CoM is aligned with the goal. In this process, terrain map and physical features contribute to the constraints for the NLP solver and significantly influences the optimized state. With the initial and final optimal stance, we find the optimal transition between them by solving another NLP problem. The transitional trajectory profile is parameterized as a spline curve and is discretized with the intermediate knots where constraints are enforced. The optimal transition logic adopts the phase-based parameterization idea from [3].

A. Optimal Stance

The optimal stance posture is affected by the environment in a great deal. In the following paper, terrain is presented in GripMap [10], a data structure with 2D arrays indicating the (x, y) location and associated height at each grid. The terrain cost function is given by

$$J_{ter}(x, y) = terrainVar(x, y) + terrainCov(x, y) \quad (1)$$

Here $terrainVar()$ returns the variance of height around a grid; low variance means the terrain location is relatively flat. $terrainCov()$ indicates the confidence of the height measurement at a spot. The robot inclines to choose areas with low uncertainty for footholds to increase the stability of stances.

Assume the system has n legs and m joints on each leg. Accordingly, for $i \in [1, n]$, the decision variable is defined as

$$\mathbf{x} = \left[\mathbf{r}_w \ \boldsymbol{\theta}_w \ \{\mathbf{q}^i\} \ \{\mathbf{f}^i\}\right] \quad (2)$$

where $\mathbf{r}_w \in \mathbb{R}^3$ and $\boldsymbol{\theta}_w \in \mathbb{R}^3$ are the CoM position and orientation of the robot represented in world frame; $\{\mathbf{q}^i \in \mathbb{R}^3\}$ is a set containing the angular position of each joint and $\{\mathbf{f}^i \in \mathbb{R}^3\}$ are vectors of forces exerted on the feet, expressed in the orthogonal projection aligned with world reference coordinates.

To figure out an optimal stance given a designated CoM position and terrain map surrounding that point, we formulate the cost function to be minimized with several considerations: The stance should (1) balance the joint efforts from each leg; (2) experience small frictional forces; and (3) have a low terrain cost. Different weights are assigned to each term:

$$\min_{\mathbf{x}} \ w_1 Var(\boldsymbol{\tau}) + w_2 \sum_i \mathbf{f}^{iT} Q \mathbf{f}^i + w_3 \sum_i J_{ter}(p_w^{ix}, p_w^{iy}) \quad (3)$$

$$\text{s.t.} \quad \mathbf{p}_c^i = FK(\mathbf{q}^i) \quad (4)$$

$$\mathbf{p}_w^i = \mathbf{R}_{wc} FK(\mathbf{q}^i) + \mathbf{t}_{wc} \quad (5)$$

$$\boldsymbol{\tau}^i = \mathbf{J}^T \begin{bmatrix} \mathbf{p}_c^i \times \mathbf{R}_{cw} \mathbf{f}^i \\ \mathbf{R}_{cw} \mathbf{f}^i \end{bmatrix} \quad (6)$$

where $Var(\boldsymbol{\tau})$ evaluates the variance of torques among all legs. $\mathbf{p}$ with subscript c and w indicates the foot position in body frame or world frame. Torque $\boldsymbol{\tau}$ and Jaconian transformation $\mathbf{J}$ is derived from forward kinematics function $FK()$ and is based on the leg's configuration. $\mathbf{R}$ and $\mathbf{t}$ with subscripts are the rotation matrix and transformation matrix from one frame to another.

Meanwhile, the system subjects to both equality and inequality constraints:

$$p_w^{iz} = h(p_w^{ix}, p_w^{iy}) \tag{7}$$

$$\begin{bmatrix} \sum_i \mathbf{f}_i \\ \sum_i (\mathbf{p}_w^i - \mathbf{r}_w) \times \mathbf{f}_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \\ 0_{3\times 1} \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & -u \\ -1 & 0 & -u \\ 0 & 1 & -u \\ 0 & -1 & -u \end{bmatrix} \mathbf{R}_{kw}^i \mathbf{f}^i \leq 0_{5\times 1} \tag{9}$$

$$\boldsymbol{\tau} < \boldsymbol{\tau}_{max} \tag{10}$$

where $h()$ gives the corresponding height of a grid. Among these constraints, Equation 7 restrains feet height to coincide with terrain height. Equation 8 requires ground supporting forces to balance the robot body weight. Equation 9 limits every ground force within the frictional cone where u is terrain frictional coefficient. $\mathbf{R}_{kw}^i$ is a rotational matrix transforming foot forces from world frame to contact frame in which origin is at the contact point and z axis is perpendicular to the contact plane. Equation 10 enforces joint torques' limit depends on working range of actuators. Since we only consider a quasi-static motion throughout the process for simplicity, the second derivative of the motion is not appended in dynamics balance. For the hexapod $i = 6$ like Mat6, this nonlinear program entails 42 decision variables, 12 equality constraints and 42 inequality constraints. It can be handled by NLP solvers like FMINCON for MATLAB and SNOPT or IPOPT for C++.

B. Optimal Transition

The methodology to find a feasible solution in optimal transition is analogous to that in optimal stance. However, NLP formulations do not naturally allow the objective function and constraints to be turned on or off arbitrarily during an iteration. In other words, we have to compose a set of continuous decision variables which can be optimized over the cycle despite the discrete gait pattern legged robots have to follow. The cycle is defined as a transformation from one stance to another, where each leg goes through one swing mode bounded by two stance modes. [3] suggests a phase-based parameterization comprising intermediate knots and phase knots as the junction for the trajectory profile curves of CoM, foot positions as well as foot forces. Comparatively, in this work we solve a minimization problem instead of a feasibility problem which focuses on eliminating unrealistic local minimum. The smooth connection of curves between the knots and phases ensures that NLP solvers can be used in our case. The constraints enforced on the knots can be approximately considered as constraints imposed on the whole curve, with negligible violation. This variable consists of times and knot values of all intermediate knots and phase knots. We take

$$\mathbf{x} = [\{\boldsymbol{\Omega}^j\}\ \{\mathbf{t}^i\}\ \{\boldsymbol{\Phi}^{ik}\}\ \{\boldsymbol{\Psi}^{il}\}] \tag{11}$$

$$\boldsymbol{\Omega}^j = [t^j\ \mathbf{r}_w(t^j)\ \boldsymbol{\theta}_w(t^j)\ \dot{\mathbf{r}}_w(t^j)\ \dot{\boldsymbol{\theta}}_w(t^j)] \qquad j \in [1, o] \tag{12}$$

$$\boldsymbol{\Phi}^{ik} = [t^{ik}\ \mathbf{p}_w^i(t^{ik})\ \dot{\mathbf{p}}_w^i(t^{ik})] \qquad k \in [1, m] \tag{13}$$

$$\boldsymbol{\Psi}^{il} = [t^{il}\ \mathbf{f}^i(t^{il})\ \dot{\mathbf{f}}^i(t^{il})] \qquad l \in [1, n] \tag{14}$$

where i represents leg index and $\{\mathbf{t}^i\} \in \mathbb{R}^2$ is a set of time instance of phase knots (at the beginning and end of the swing mode). $\boldsymbol{\Omega}^j \in \mathbb{R}^{13}$ is the j-th intermediate knot of CoM positions' and Euler angles' trajectory spline while $\boldsymbol{\Phi}^{ik} \in \mathbb{R}^7$ and $\boldsymbol{\Psi}^{il} \in \mathbb{R}^7$ are the k-th and l-th intermediate knots of foot positions' and forces' trajecotry spline. o, m and n are the number of total intermediate knots of each parameterization respectively. To connect these knots, we adopt the cubic Hermite splines interpolation method to define the polynominal.

For two ajacent knots $(0, x_0, \dot{x}_0)$ and $(1, x_1, \dot{x}_1)$ with first-order derivative present, we could smoothly connect two knots and compute the curve value anywhere between $t \in [0, 1]$ by

$$x(t) = (2t^3 - 3t^2 + 1)x_0 + (t^3 - 2t^3 + t)\dot{x}_0 + (-2t^3 + 3t^2)x_1 + (t^3 - t^2)\dot{x}_1 \tag{15}$$

This method can be easily generalized to the multidimensional curve and the arbitrary time interval. We assign T as the total length of time in a cycle. In such layout, three types of knots are needed to fully define a cubic spline. Boundary knots for $t = 0$ and $t = T$: $\mathbf{r}(0)$, $\mathbf{p}^i(0)$ and $\mathbf{f}^i(0)$ corresponds to their initial configurations while values at $t = T$ are given from the target posture. What's more, at phase knots, foot forces, positions and their derivatives must be zero as shown in Fig. 2.

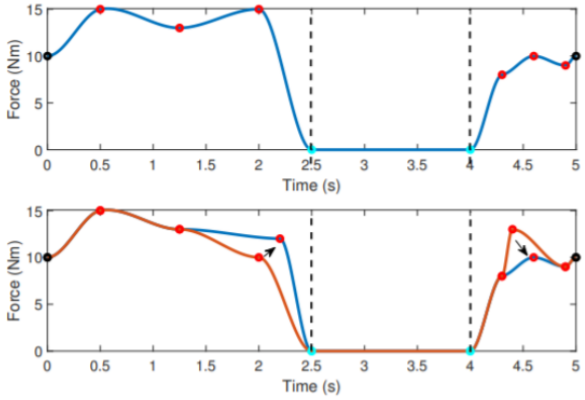


Fig. 2. An example foot force trajectory profile. **Top**: Blue line is the force spline curve. Red circles are locations of spline intermediate knots. Black circles are boundary knots. Cyan circles are phase knots. Black dashlines indicate the leg mode transition time instances. The total time of the motion plan is 5s, and mode transitions happen at 2.5s and 4s. The force profile has 6 intermediate knots. **Bottom**: Adjusting intermediate knots results in different continuous curves in optimization.

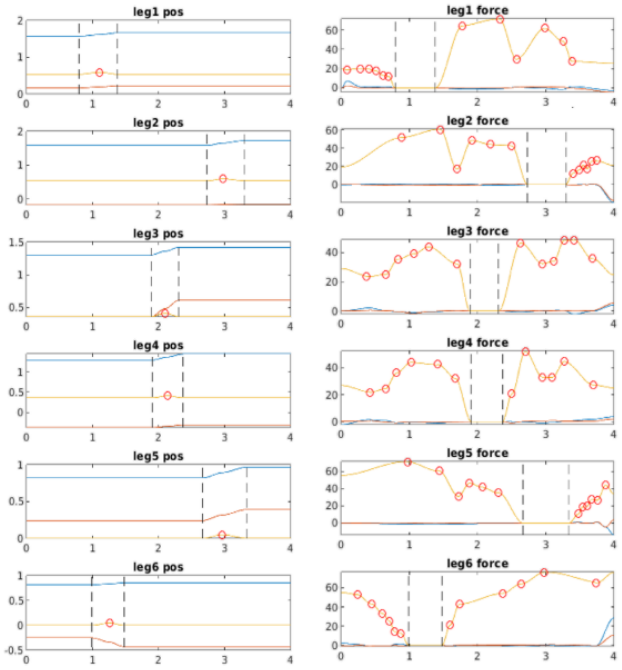


Fig. 3. Phase-based parameterization of feet motion $\{\mathbf{p}^i\}$ and force $\{\mathbf{f}^i\}$ in a simulation. Each phase (swing and stance) is represented by either a constant value or a sequence of cubic polynomials with continuous derivatives at the junctions. Red circle dots represents the intermediate knots either of position (**Left**) in swing mode or forces (**Right**) in stance mode. Dashlines are the switching time after the optimization on the gait pattern.

Other approaches to interpolate the nodes using lower-degree implicit integration rules like Euler's rules or trapezoidal method, throw down the computational complexity on one side, whereas moderate the accuracy of spline on the other side. The shortcoming in accuracy of these approaches can be compensated by a lower sparsity of nodes, but reciprocally, increment in knots burdens the computation. Overall, it is a compromise between the discretization and efficiency of the solver to be made in choosing an ideal interpolation tool.

The objective function enforcing optimality on knots is defined as

$$\min_{\mathbf{x}} \; w_1 \sum_i \mathbf{\Psi}^{iT} \mathbf{Q} \mathbf{\Psi}^i + w_2 J\big(\mathbf{\Phi}, {}^*h(x,y)\big) \tag{16}$$

reflecting a tradeoff between the first term minimizing ground supporting forces and the second term $J()$ penalizing swing position below a certain level from the terrain to avoid the collision. ${}^*h(x,y)$ is the pointer function to acquire the terrain height at a certain position.

Optimal transition grabs the same constraints from that in optimal stance, appended by the restriction on knots time, which should be positive but not exceed the execution time. In our optimal transition program, we assign $i = 6$, $m = 2$, $n = 6$, $o = 6$ and $N = 25$. The system scales 356 decision variables, 150 nonlinear equality constraints, and 626 linear inequality constraints.

III. Iterative Linear Quadratic Regulator Approach

In this section, we present iLQR method in solving the TO problem on legged systems with contact constraints. The robot is approximately modeled in a single rigid body dynamics (SRBD) formulation and an implicit contact model, which incorporates the contact force vector into control input. In this model, the contact constraint can be directly imposed as a control-only constraint in LQR setting, and be solved in a recursive Riccati approach. Compared to Direct Collocation method, iLQR requires the gait pattern to be pre-specified yet a low-level controller is no longer needed.

A. Dynamics Model

We define a implicit contact model where contact force vector is entailed in the control input rather than is a function of other variables. The benefits of using the implicit contact model over the explicit contact model is detailed in [8]. In SRBD model, we describes the free-floating body along with the kinematics for each leg in a form of

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}}_w \\ \dot{\boldsymbol{\theta}}_w \\ \ddot{\mathbf{r}}_w \\ \dot{\boldsymbol{\omega}}_b \\ \{\dot{\mathbf{p}}^i_w\} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_w \\ \mathbf{T}(\boldsymbol{\theta})\boldsymbol{\omega}_b \\ \mathbf{g} + \frac{1}{m}\sum_i c^i \mathbf{f}^i \\ \mathbf{I}^{-1}(-\boldsymbol{\omega}_b \times \mathbf{I}\boldsymbol{\omega}_b + \sum_i (\mathbf{p}^i_w - \mathbf{r}_w) \times c^i \mathbf{f}^i) \\ \{\dot{\mathbf{p}}^i_w\} \end{bmatrix} \tag{17}$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = f\left(\begin{bmatrix} \mathbf{r}_w \\ \boldsymbol{\theta}_w \\ \dot{\mathbf{r}}_w \\ \boldsymbol{\omega}_b \\ \{\mathbf{p}^i_w\} \end{bmatrix}, \begin{bmatrix} \{\dot{\mathbf{p}}^i_w\} \\ \{\mathbf{f}^i\} \end{bmatrix} \right) \tag{18}$$

where $\mathbf{T}$ is the rotational matrix from angular velocities in the base frame to the Euler angles derivatives in the global frame, $\mathbf{g}$ is the gravitational acceleration in the world frame, $\mathbf{I}$ and m are the moment of inertia about the CoM and the total mass respectively. The inertia is assumed to be constant and taken at the default configuration of the robot. $\boldsymbol{\omega}_b$ is the angular rate of the body. c^i denotes whether the leg touches the ground or not, which gives 1 if does and 0 otherwise.

We assume that the end-effector in contact with the ground has no slippage occurred. The control-only constraints can be expressed as

$$c^i \dot{\mathbf{p}}^i_w = 0 \tag{19}$$

since when the swing foot is away from the ground, c^i equals to zeros and the left-hand side of the expression is zero for granted. While the foot stays stationary, the position derivative is zero and the equation still holds.

Another approach to construct the model substitutes foot positions with joint angles in the decision variable. Though joint angles are direct parameters to control the robot's actuators, a more complex format of constraints comes with ensuing defects. A comprehensive comparison between the two approaches and the reason why the latter overwhelms the former are illustrated at the end of the section.

B. Problem Formulation

In this section, we assume a non-linear discrete system of

$$\mathbf{x}(n+1) = \mathbf{A}(n)\mathbf{x}(n) + \mathbf{B}(n)\mathbf{u}(n) \tag{20}$$

which is integrated from the differential form of the state equation and is discretized by interpolation on continuous $\mathbf{x}(t)$ and $\mathbf{u}(t)$. $\mathbf{u}(n)$ is the control policy where $n \in \{0, \ldots, N-1\}$. Throughout the following part, total time of one gait cycle is discretized by $N-1$ time steps, each of which spans $\delta t = T/(N-1)$. $\mathbf{A}(n)$ and $\mathbf{B}(n)$ are obtained from linearizing around each time step along the nominal trajectory. The control input is optimized through altering time-varying feedback and feedforward controller by

$$\mathbf{u}(n) = \mathbf{u}_{ref}(n) + \mathbf{u}_{ff}(n) + \mathbf{K}(n)\big(\mathbf{x}(n) - \mathbf{x}_{ref}(n)\big) \tag{21}$$

where $\mathbf{K}(n)$ and $\mathbf{u}_{ff}(n)$ are control gain matrix and feedforward control action. $\mathbf{u}_{ref}(n)$ is the reference state, in the vicinity of which newly locally-optimal control is to be found.

Compared to SLQ formulation, minimizing the cost function in a quadratic layout could notably increase the convergence rate. In the following algorithm shown, the cost function is assumed to be

$$J = \frac{1}{2}\sum_{n=0}^{N-1}\big(\mathbf{x}(n)^T\mathbf{Q}\mathbf{x}(n) + \mathbf{u}(n)^T\mathbf{R}\mathbf{u}(n)\big) + \frac{1}{2}\big(\mathbf{x}_{ref} - \mathbf{x}(N)\big)^T\mathbf{Q}_f\big(\mathbf{x}_{ref} - \mathbf{x}(N)\big) \tag{22}$$

where $\mathbf{x}_{ref}$ is same the target state calculated by the optimal stance. Q, R and Q_f are weightings for intermediate cost, input cost and final cost.

C. Algorithm

Inside every iteration of the proposed iLQR algorithm, we solve a constrained, finite-horizon LQR problem, which updates an optimal control solution with a new cost. The LQR problem is defined in the system dynamics governed by Equation 17 and 18 and constrained by Equation 19. [9] proposed the method of using distinctive Lagrange Multipliers to handle the system dynamics, state-control, and state-only constraints respectively and then solving the discrete Bellman equation. In this work, we refer to the same structure but ignore the state-only constraint.

The algorithm begins by drawing an optimal control from the first-order necessary condition subject to a set of constraints, expressed in linear equations. This is attained by first solving Riccati equations and then propagating forward to find a co-state vector sequence $\boldsymbol{\lambda} = [\boldsymbol{\lambda}(0), \ldots, \boldsymbol{\lambda}(N)]$, and a multiplier vector sequence $\boldsymbol{\mu} = [\boldsymbol{\mu}(0), \ldots, \boldsymbol{\mu}(N-1)]$, associated with controls and state-control constraints respectively. Finally, the procedure is wrapped up by a feedforward pass to derive a full spectrum of feedforward control policy and state setting off from the initial boundary condition towards the end.

LQR algorithm iteratively implemented to find optimal trajectory gives iLQR. In iLQR, we start with a guessing trajectory, which we assume the robot is standing still throughout the duration. Given a reference trajectory, we continue by refining the guess until the optimal trajectory converges in a convex set. In each iteration, we linearize around the trajectory and then execute Algorithm 1 to obtain the feedforward control law, control gain, and new state sequence. A cost is computed afterward given final state and each step of the updated trajectory. Since iLQR might have the overshoot problem, the line-search learning process is necessary to find the local minimum. The line-search parameter α controls the maximum rate to move along the feedforward component of the control law update. Starting with a full rate in the descent direction ($\alpha = 1$), this parameter progressively shrinks by a certain fraction α_d until the cost associated with the updated controller input is lower than the cost of full learning rate.

Algorithm 1 LQR Algorithm with Constraints

Require:
- System Dynamics: $\mathbf{x}(n+1) = \mathbf{A}(n)\mathbf{x}(n) + \mathbf{B}(n)\mathbf{u}(n)$
- Constraints: $\mathbf{D}(n)\mathbf{u}(n) = 0$
- Cost Function:
  $J = \frac{1}{2}\sum_{n=0}^{N-1}\big(\mathbf{x}(n)^T\mathbf{Q}\mathbf{x}(n) + \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t)\big) +$
  $\frac{1}{2}(\mathbf{x}_{ref} - \mathbf{x}(N))^T\mathbf{Q}_f(\mathbf{x}_{ref} - \mathbf{x}(N))$

for $n = 0 : N-1$ **do**
  $\hat{\mathbf{D}}(n) = [\mathbf{D}(n)\mathbf{R}^{-1}\mathbf{D}(n)^T]^{-1}$
  $\hat{\mathbf{R}}(n) = \mathbf{B}(n)\mathbf{R}^{-1}[\mathbf{I} - \mathbf{D}(n)^T\hat{\mathbf{D}}(n)\mathbf{D}(n)\mathbf{R}^{-1}]\mathbf{B}(n)^T$
end for

- Backward Pass
$\mathbf{P}(N) = \mathbf{Q}_f$
$\mathbf{s}(N) = -\mathbf{Q}_f(\mathbf{x}(N) - \mathbf{x}_{ref})$ ▷ Boundary Conditions
for $n = N-1 : 0$ **do**
  $\mathbf{M}(n) = [\mathbf{I} + \hat{\mathbf{R}}(n)\mathbf{P}(n+1)]^{-1}$
  $\mathbf{P}(n) = \mathbf{Q}^T + \mathbf{A}(n)^T\mathbf{P}(n+1)\mathbf{M}(n)\mathbf{A}(n)$
  $\mathbf{s}(n) = \mathbf{A}(n)^T\mathbf{M}(n)^T\mathbf{s}(n+1)$
end for

- Forward Pass
$\mathbf{x}(0) = \mathbf{x}_0$ ▷ Boundary Condition
for $n = 0 : N-1$ **do**
  $\mathbf{v} = -\mathbf{M}(n)\hat{\mathbf{R}}(n)\mathbf{s}(n+1)$
  $\mathbf{x}(n+1) = \mathbf{M}(n)\mathbf{A}(n)\mathbf{x}(n) + \mathbf{v}$
  $\boldsymbol{\lambda} = \mathbf{P}(n+1)\mathbf{x}(n+1) + \mathbf{s}(n+1)$
  $\boldsymbol{\mu} = -\hat{\mathbf{D}}(n) - \mathbf{D}(n)\mathbf{R}^{-1}\mathbf{B}(n)^T\boldsymbol{\lambda}$
  $\mathbf{u}(n) = -\mathbf{R}^{-1}[\mathbf{B}(n)^T\boldsymbol{\lambda} + \mathbf{D}(n)^T\boldsymbol{\mu}]$
  $\mathbf{K}(n) = -\mathbf{R}\mathbf{B}(n)^T\mathbf{P}(n+1)\mathbf{M}(n)\mathbf{A}(n) -$
  $\mathbf{R}^{-1}\mathbf{D}^T(n)\hat{\mathbf{D}}(n)\mathbf{D}(n)\mathbf{R}^{-1}\mathbf{B}(n)^{-1}\mathbf{P}(n+1)\mathbf{M}(n)\mathbf{A}(n)$
  $\mathbf{u}_{ff}(n) = \mathbf{R}^{-1}[-\mathbf{B}(n)^{-1}(\mathbf{P}(n+1)\mathbf{v} + \mathbf{s}(n+1)) +$
  $\mathbf{D}(n)^{-1}\hat{\mathbf{D}}(n)\mathbf{D}(n)\mathbf{R}^{-1}\mathbf{B}(n)^{-1}(\mathbf{P}(n+1))\mathbf{v} + \mathbf{s}(n+1))$
end for
return
  $\mathbf{x}(n)$ where $n \in \{0, \ldots, N\}$
  $\mathbf{u}_{ff}(n), \mathbf{K}(n)$ where $n \in \{0, \ldots, N-1\}$

Algorithm 2 iLQR Algorithm

- Initialize a stable control and state trajectory. We suppose the robot is standing still in initialization.
  $\mathbf{x}(n)$ where $n \in \{0, \dots, N\}$
  $\mathbf{u}(n)$ where $n \in \{0, \dots, N-1\}$
repeat
  - Linearize the system along the previous (reference) trajectory. Suppose $f = \mathbf{A}(t)\delta\mathbf{x}(t) + \mathbf{B}(t)\delta\mathbf{u}(t)$
    $\mathbf{A}(n) = \frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_{ref}(n)}\delta t + \mathbf{I}$
    $\mathbf{B}(n) = \frac{\partial f}{\partial \mathbf{u}}|_{\mathbf{u}=\mathbf{u}_{ref}(n)}\delta t$
  - Execute **Algorithm 1** which returns
    $\mathbf{x}(n)$ where $n \in \{0, \dots, N\}$
    $\mathbf{u}_{ff}(n), \mathbf{K}(n)$ where $n \in \{0, \dots, N-1\}$
  - Compute the cost function
    $J = \frac{1}{2}\sum_{n=0}^{N-1}\left(\mathbf{x}(n)^T\mathbf{Q}\mathbf{x}(n) + \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t)\right) +$
    $\frac{1}{2}(\mathbf{x}_{ref} - \mathbf{x}(N))^T\mathbf{Q}_f(\mathbf{x}_{ref} - \mathbf{x}(N))$
  - Initialize the line search
    $\alpha = 1$
  repeat
    - Under the control:
      $\mathbf{u}(n) = \mathbf{u}_{ref}(n) + \alpha\mathbf{u}_{ff}(n) + \mathbf{K}(n)(\mathbf{x}(n) - \mathbf{x}_{ref}(n))$
    - Forward simulate the system dynamics
      $\mathbf{x}(n)$ where $n \in \{0, \dots, N\}$
      $\mathbf{u}(n)$ and $\mathbf{u}_{ff}(n)$ where $n \in \{0, \dots, N-1\}$
    - Compute the new cost
      $J = h(\mathbf{x}(N)) + \sum_0^{N-1} l(\mathbf{x}(n), \mathbf{u}(n))dn$
    - Decrease α by a constant α_d
      $\alpha = \alpha/\alpha_d$
  until lower cost found or the maximum number of line search steps reached
until the maximum number of iterations achieved or the cost converges

The optimal control we draw from the algorithms above excludes the influence of terrain. Practically, in a sense that we assume robot legs to be massless, it is feasible to change the trajectory of swing legs on the fly to avoid obstacles, without interfering with the dynamics of the floating body and stance legs. Considering the obstacles avoidance separately exempts the state-only constraint term and simplifies the Riccati equation.

D. Comparison

In this contribution, we compare two different system dynamics parameterizations in terms of their convergence rates. Apart from the aforementioned foot position-based setup, the other more straightforward parameterization takes an executable input comprised of joints' angular speeds.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}}_w \\ \dot{\boldsymbol{\theta}}_w \\ \ddot{\mathbf{r}}_w \\ \dot{\boldsymbol{\omega}}_b \\ \{\ddot{\mathbf{q}}^i\} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}}_w \\ \mathbf{T}(\boldsymbol{\theta})\boldsymbol{\omega}_b \\ \mathbf{g} + \frac{1}{m}\sum_i c^i\mathbf{f}^i \\ \mathbf{I}^{-1}(-\boldsymbol{\omega}_b \times \mathbf{I}\boldsymbol{\omega}_b + \sum_i(\mathbf{p}_w^i - \mathbf{r}_w) \times c^i\mathbf{f}^i) \\ \{\dot{\mathbf{q}}^i\} \end{bmatrix} \tag{23}$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = f\left(\begin{bmatrix} \mathbf{r}_w \\ \boldsymbol{\theta}_w \\ \dot{\mathbf{r}}_w \\ \boldsymbol{\omega}_b \\ \{\mathbf{q}^i\} \end{bmatrix}, \begin{bmatrix} \{\dot{\mathbf{q}}^i\} \\ \{\mathbf{f}^i\} \end{bmatrix}\right) \tag{24}$$

Note that this definition is fundamentally similar to Equation 17 and 18, both formatted in SRBD, except that the last set of terms are represented in joints' Euler angles. This brings the benefit that inverse kinematics is no more necessary, averting plural leg configurations it might potentially yield. However, it becomes more complicated in respect of the constraints expression

$$\frac{\partial \mathbf{p}^i}{\partial \mathbf{r}}\dot{\mathbf{r}} + \frac{\partial \mathbf{p}^i}{\partial \boldsymbol{\theta}}\dot{\boldsymbol{\theta}} + \frac{\partial \mathbf{p}^i}{\partial \mathbf{q}}\dot{\mathbf{q}} = 0 \tag{25}$$

which can be interpreted as the derivative form of $\mathbf{p}^i(\mathbf{r}, \boldsymbol{\theta}, \mathbf{q}) = 0$, where $\mathbf{p}$ is obtained from the forward kinematics. Unfortunately, it turns out to be not just a control-only constraint but a mixed state-control constraint. As a result, the Lagrange Multiplier expression is also subject to the state constraint term and Algorithm 1 should be modified accordingly.

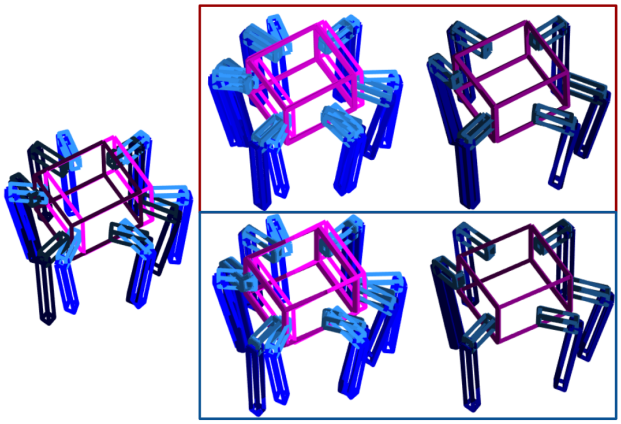


Fig. 4. The robot is programmed to move forward on a 3-3 gait pattern. **Left**: The initial pose (light) and the target pose (dark); **Middle Column**: Evolution of each iteration in iLQR (from light to dark); **Right Column**: Comparison between the final pose and the target pose. The Upper red box contains the joint angles' parameterization, which runs 7 iterations before convergence, while the bottom blue box is parameterized in foot positions, which needs only 3 iterations before convergence. Total runtime of the former is 1.9 times faster than the latter using MATLAB ode45 solver.

Fig. 4 shows that both configurations can reach a stable final pose with fine error to the target. However, we see foot position parameterization converges nearly 2 times faster than that in joint angles since fewer iterations are run to convergence. Simulations display that 90% of time for each iteration is consumed on solving for the propagation along the trajectory. Therefore, fewer iterations and line-searching loops would significantly lower the total duration to find an optimal solution. When TO runs online, we hope to keep the optimization runtime short. A quick convergence rate implies that robots could react instantly, which is especially demanding in dynamic environments. In such, iLQR approach with

frequent updating rate can be used in an MPC fashion during the trajectory execution to prevent the disturbance in terrain map measurement, state estimation, and tracking execution.

IV. Experiment

A. System Overview

Fig. 5 gives an overview on the system pipeline which comprises the main flow colored in blue and auxiliary blocks colored in crimson. After the user specifies a goal (x, y) on the elevation map, the planner will generate an intermediate pose towards the user-specified goal. The distance between the intermediate pose and the current pose is a fixed value of one step for each foot. Next, the robot solves an optimization problem online to select a set of footholds on the terrain map. The initial state and final state are then transferred to the boundary condition for a direct collocation optimizer, which computes optimal trajectories for body pose, 6-foot positions, and 6-foot ground reaction forces. Finally, the low-level controller converts the whole-body motion trajectory into joint angle trajectories and torque trajectories to command motors. The body balance controller generates additional controls to balance the body.

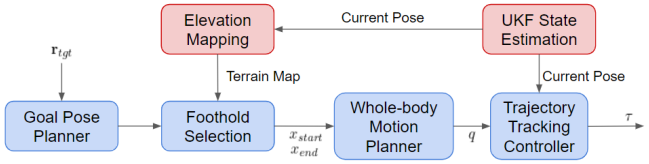


Fig. 5. Block architecture of the optimization-based controller and its relationship with state estimation and elevation mapping.

1) State Estimation: The state estimator evaluates the pose of COM in the inertial frame as well as the robot's joint angles by extracting information from proprioceptive sensors. The common approach by estimating the displacement and velocity of the legged robot through onboard IMU and joint encoders is a lack of accuracy since some parameters like yaw orientation is no way observable. Therefore, adding visual sensors is a complementary measure to reach better precision. We mount two Intel RealSense D435i cameras to help with the localization and mapping. D435i is an RGBD camera that provides visual sensing as well as the depth measurement through a point cloud. In our state estimation setting, we apply Unscented Kalman Filter (UKF) [11] to fuse multiple odometry together, including the leg odometry, Visual Inertial Odometry (VIO) [12], and IMU.

2) Elevation Map: To achieve better terrain adaptability, we need to generate a clear local map around the robot to enable foothold planning and trajectory generation. Local map construction is considered a solved problem in the SLAM community, however on legged robots, it is different in many ways and should be treated specially. Therefore, two visual sensors are mounted in front of the robot at distinctive positions and pitch angles to provide a wide field of view. One camera looks horizontally forward while the other tilts downwards to cover the near end.

The elevation map is a robot-centric grid map [13] representing the height of each grid in a 2D array. It updates every time after a new estimated pose or the point cloud data comes in, giving an up-to-date elevation as well as the trust boundary of the terrain.

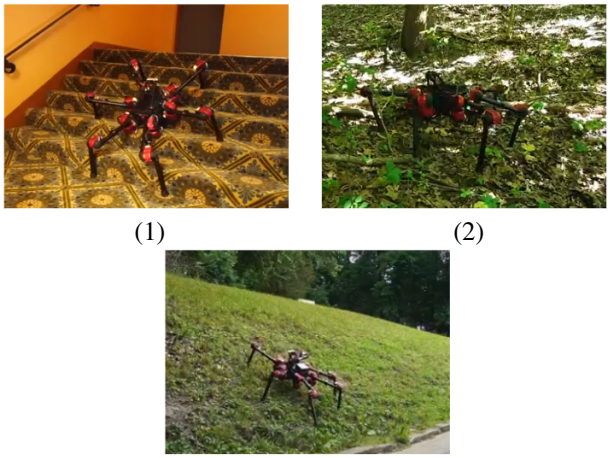


Fig. 6. Real world testing under scenarioa: (1) the staircase (each step is around 18 cm wide and 28 cm high); (2) jungle lands; (3) the 30° slope.

B. Results

We perform tasks on assorted terrains, among which stairs climbing is an epitome to exhibit the robot's ability to walk on challenging terrains. The climbing is demonstrated on a staircase in the US standard of a 7 in (17.8 cm) rise and 11 in (27.9 cm) run. The mapping of the staircase is mostly completed through sensors on the robot being placed at the flat platform before the staircase, refreshed concurrently during the climbing. The trajectory of each gait is an outcome of Direct Collocation optimization, executed by low-level tracking controllers. The gait is set to be a conservative 2-2-2 pattern, denoting 2 legs on the fly for each time. If the real map mismatches that from sensing, the robot will manage to regain balance by descending the dangling leg until it touches

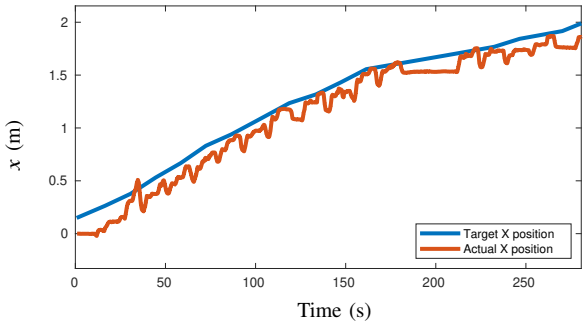


Fig. 7. The target (blue) and feedback (orange) pose in x direction during stair climbing. In our setting, x-axis is pointing forward to the rise of the stiarcase with z-axis pointing upward. The feedback position keeps stable for a period of time at around the 200th second since the robot pauses for 30 seconds.

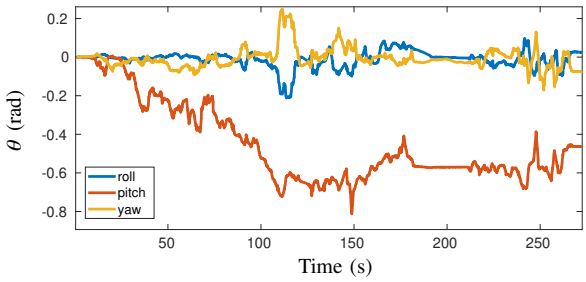


Fig. 8. The values of roll (blue), pitch (orange) and yaw (yellow) of the body throughout climbing. The roll and yaw angles fluctuates around zero in most of time while the pitch angle increases when robot ascends.

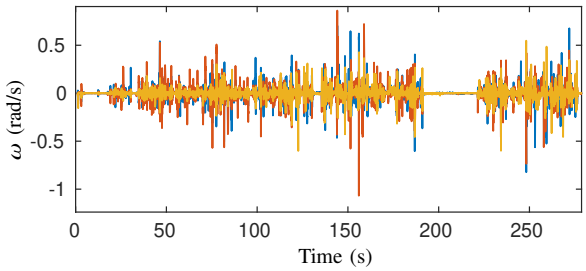


Fig. 9. The angular velocities of the robot measured by onboard IMU. Blue, orange and yellow lines feature angular velocities about the x, y, and z axes respectively. The peak magnitude records 1.068 rad/s.

the ground. This might happen when the planning foothold is on the verge of the stair where there is an abrupt fall on height. Similarly, when the foot sensor or the joint encoder detects the unexpected collision, the robot will stop the leg from further poking into the ground.

The video clip showing the robot climbing a whole flight of stairs has been uploaded[1]. In testing, the robot starts from the flat platform at time 0 and reaches the top of the staircase at around the 250th second, with a total displacement of 2 meters in forward direction. In Fig. 7, we compare the feedback position and the target on x axis, and see a smooth tracking of movement despite small fluctuation in motion caused by the robot's stance legs switching. Fig. 8 and Fig. 9 shows the orientation and angular velocity of the robot. Though we do not have access to the ground truth of robot orientation, the estimation value approximately reflects the real motion of the robot. It can be seen that among all axes, the maximum value of angular velocity is 1.068 rad/s, which turns out to be a relatively small margin and implies the fluctuation is controllable.

We later exploit its ability in climbing by deploying it on the slope with the maximum incline. The test has been run on a grassland slope with a 40-degree pitch. To our best knowledge, this is the largest incline the legged system has ever successfully climbed.

V. Conclusion

Direct Collocation approach adopted in this paper formulates a whole-body trajectory restricted by the terrain and physical constraints. We optimize over the discrete gait pattern using a set of continuous decision variables. The hierarchical control structure separating the foothold selection and transition planner has proven effective in both simulation and real-robot testing. iLQR approach, alternatively, provides a faster solution and offers a feasible control policy together with an optimal trajectory. Future works include applying iLQR in MPC fashion to reject disturbance in measurements.

[1] https://youtu.be/98wdAfD1_2w

VI. Acknowledgement

I would like to thank Professor Jungwon Seo, and my home university, The Hong Kong University of Science and Technology for their support of my participation in the RISS program. Besides, I want to thank the RISS 2020 cohort and the RI community to make this program vibrant and enriching. Finally, this whole program would not happen due to the COVID-19 without the continuing commitment from directors Dr. John Dolan and Ms. Rachel Burcin.

References

[1] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[2] D. Pardo, M. Neunert, A. Winkler, R. Grandia, and J. Buchli, "Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion," *Procceeding of Robotics Science and System*, 2017.

[3] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[4] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwel, "Proceedings of the institution of mechanical engineers. part i, journal of systems and control engineering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 256 (6), 2011, pp. p.831–849.

[5] D. Pardo, M. Neunert, A. W. Winkler, and J. Buchli, "Projection based whole body motion planning for legged robots," unpublished.

[6] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, "Models, feedback control, and open problems of 3d bipedal robotic walking," *Automatica*, vol. 50, pp. 1955–1988, 2014.

[7] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, 2017.

[8] M. Neunert, F. Farshidian, and J. Buchli, "Efficient whole-body trajectory optimization using contact constraint relaxation," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 43–48.

[9] A. Sideris and L. A. Rodriguez, "A riccati approach to equality constrained linear quadratic optimal control," in *Proceedings of the 2010 American Control Conference*, 2010, pp. 5167–5172.

[10] P. Fankhauser and M. Hutter, *A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation*. Cham: Springer International Publishing, 2016, pp. 99–120.

[11] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, "The control toolbox — an open-source c++ library for robotics, optimal and model predictive control," *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 123–129, 2018.

[12] S. Yang, H. Kumar, Z. Gu, X. Zhang, M. Travers, and H. Choset, "State estimation for legged robots using contact-centric leg odometry," 2019.

[13] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

On the Value of Modeling Dependencies in Weak Supervision

Salva Rühling Cachay[1], Benedikt Boecking[2] and Artur Dubrawski[2]

***Abstract*— Large sets of labeled data are key in supervised machine learning methods. However, labeling by hand is expensive, both in terms of time and cost, creating a major bottleneck when deploying such methods. Without access to ground truth labels, weak supervision is a promising alternative that overcomes this issue by combining the outputs of a set of noisy, user-given heuristics. These heuristics may conflict, overlap, and have complex relationships. There has been extensive work on modeling and learning these dependencies, yet the general value of modeling dependencies to improve downstream model performance has not been studied across a wide variety of scenarios. In a controlled, experimental study, we explore this gap and characterize the settings in which modeling dependencies boosts end-model performance and quantify the expected performance gain. Surprisingly, we find that modeling seemingly sensible dependencies can significantly *deteriorate* performance by up to 4 AUC points. Through this contribution we hope to enhance the decision-making of practitioners as we show that ignoring potential dependencies is a reasonable baseline in many cases.**

***Index Terms*— weak supervision, machine learning, graphical models**

I. INTRODUCTION

The success of supervised machine learning methods relies on the availability of large amounts of labeled data. The common process of manual data annotation by humans, especially when domain experts need to be involved, is expensive, both in terms of time and cost, and as such presents a major bottleneck for deploying supervised learning methods to new domains and applications.

Recently, data programming, a framework that makes use of multiple weak supervision sources, has emerged as a promising alternative to manual data annotation [1]. In this framework, users need to encode domain knowledge into so-called *labeling functions* (LF), the weak supervision sources, such as domain heuristics (as in Fig. 1), knowledge bases, or pre-trained classifiers that each noisily label a subset of the data. Using multiple of such sources the framework learns a generative model of the sources and the latent true label. One can then use the learned model to estimate *probabilistic* labels, which are usually used to train a *downstream model* (also denoted as end-model), replacing the need to obtain ground truth labels by manual annotation of individual samples.

In practice, the sources of weak labels often exhibit statistical dependencies amongst each other, such as sources operating on the same or similar input, or sources deliberately defined by the user to reinforce or fix (as in Fig. 1) other, less precise, sources (some examples can be found in Table II). For this work we use the data programming framework as introduced in [1], where the premise is that experts can model any higher-order dependency. We extend it by *negated*, *bolstering*, *priority* dependencies, e.g. the latter encoding the notion that one source's vote should be prioritized over the one from a noisier source. Newer label models and model fitting approaches such as [2], [3] often only allow for pairwise correlation dependencies to be modeled.

```
def weak_source1(text_sample):
    return POSITIVE if "recommend" in text_sample else ABSTAIN
```

```
def weak_source2(text_sample):
    return NEGATIVE if "wouldn't recommend" in text_sample else ABSTAIN
```

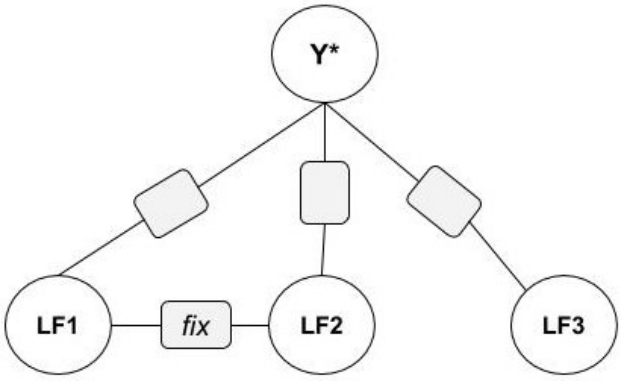


Fig. 1. Two examples for labeling functions for a sentiment analysis task and the corresponding toy factor graph (with a not shown, conditionally independent, third labeling function). Here, it is natural to model a *fixing* dependency between LF1 (top) and LF2 (middle). But, does it really improve end-model performance? If, yes. how much? Or do the LFs alone provide enough evidence to accurately estimate the latent label Y^*?

As manually providing the dependencies to be modeled does not scale and is prohibitive in development, past works present methods to automatically detect these dependencies, i.e. learn the underlying structure between the weak supervision sources [4] [5] [6].

The value to downstream model performance of the additional step of modeling dependencies is, however, not well understood. In particular, we aim to identify and quantify the impact of ignoring dependencies — a common practice in popular libraries for practical applications [7] [8] as well as related research [9] [10] — on the downstream model performance. To address this gap, we characterize the settings and circumstances under which modeling dependencies helps and

[1]Salva Rühling Cachay is with the Computer Science Department, Technical University of Darmstadt, Germany salvaruehling@gmail.com

[2]Benedikt Boecking and Artur Dubrawski are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA {boecking, awd}@andrew.cmu.edu

those in which it does not, as well as quantify, for the former, the lift in end-model performance that can be expected.

In experiments on synthetically generated data, in which we have full control and knowledge over the number of dependencies and their strength, we find that in settings with uninformative, strongly dependent sources, ignoring them leads to a loss of up to 6 ROC-AUC points in downstream performance. On the IMDB movie review dataset we, on the other hand, often observe marginal or worsening impacts. In particular, we show that modeling seemingly sensible dependencies — in highly relevant settings for practitioners — can, perhaps surprisingly, *deteriorate* downstream performance significantly by up to 4 ROC-AUC points.

The *boosting*, *marginal* or even *worsening* effects that modeling dependencies between weak supervision sources can have on the downstream model performance further underscores the importance of our presented work to gain, and give practitioners, a better understanding of the peculiarities and pitfalls that can arise.

II. RELATED WORK

Recently, the data programming paradigm was introduced, which allows users to programmatically label data through labeling functions by treating the true label as a latent variable of a generative model [1]. It was later extended to an end-to-end, open-sourced system [11].

Novel methods for solving for the parameters of the model only support the modeling of pairwise correlations [2] [3] — as such, losing some of the expressivity of [1] regarding different dependency types between weak supervision sources. The former extends data programming to the multi-task setting by exploiting the graph structure of the inverse covariance matrix among the sources [2] — in particular the fact that an entry is zero when there is no edge between the corresponding sources in the graphical model [12]. The latter finds a closed-form solution for a class of binary Ising models by using triplet methods [3].

In order to automatically learn the structure between these sources, previous work optimizes the marginal pseudolikelihood of the noisy labels [4], or makes use of robust PCA to denoise the inverse covariance matrix of the sources labels into a graph structured term [5]. A different approach, infers the structure through static analysis of the weak supervision sources code definitions and thereby reduces the sample complexity for learning the structure [6].

III. PROBLEM SETUP

In this work we make use of the data programming framework as introduced in [1]. That is, we model the weak supervision sources and the latent true label as variables of a factor graph, an expressive type probabilistic graphical model in which we can model dependencies of various types between the variables, such as a *fixing* or *reinforcing* dependency (Fig. 1 shows a simple factor graph). We focus on the binary case since the points we make only depend on the number of sources and their relationships but not on the possible labels.

Formally, given a set of n data points $\{x_i\}_{i=1}^n$ and m sources of weak supervision $\{\lambda_j\}_{j=1}^m$ we construct the label matrix Λ, where $\Lambda_{i,j} = \lambda_j(x_i) \in \{-1, 0, 1\}$ and 0 means that the source refrained from labeling the data point to any class. Only using Λ, our goal is to obtain probabilistic labels, the estimates of the latent, true labels $Y^* = \{y_i^*\}_{i=1}^n \in \{-1, 1\}^n$.

As said, we model the joint distribution $p_\theta(\Lambda, Y^*)$ as a factor graph. To this end we use factor types that represent the labeling propensity and accuracy:

$$\phi_{i,j}^{Lab} = \mathbb{1}\{\Lambda_{i,j} \neq 0\} \tag{1}$$

$$\phi_{i,j}^{Acc} = \mathbb{1}\{\Lambda_{i,j} = y_i^*\}, \tag{2}$$

as well as the dependency types *similar*, *fixing*, *reinforcing* that we inherit from [1] and, our own types by which we extend it, *negated*, *bolstering*, *priority*, e.g. for the latter:

$$\phi_{i,j,k,1}^{Prio} = \mathbb{1}\{\Lambda_{i,j} = -y_i^* \wedge \Lambda_{i,k} = y_i^*\} \tag{3}$$

$$\phi_{i,j,k,2}^{Prio} = -\mathbb{1}\{\Lambda_{i,j} = y_i^* \wedge \Lambda_{i,k} = -y_i^*\}, \tag{4}$$

which encodes our understanding that the vote of λ_k should be given priority over λ_j when both differ (for the other definitions see the appendix). For a given data point x_i let $\phi_i(\Lambda, Y^*) \in \{-1, 0, 1\}^M, M \geq 2m$ denote the concatenated vector of these factors for all sources $j = 1, \ldots, m$ and all $k \in \{1, \ldots, m\}$ for which a dependency shall be added.

We can then express the joint distribution as follows:

$$p_\theta(\Lambda, Y^*) = \frac{1}{Z_\theta} \exp\left(\sum_{i=1}^{n} \theta^T \phi_i(\Lambda, y_i^*)\right),$$

where $\theta \in \mathbb{R}^M$ is the corresponding parameter vector and Z_θ is a normalizing constant. We note that other, arbitrary dependency types can be easily modeled — the types above, however, occur most commonly in practice.

Given the label matrix Λ we can now learn the parameter $\hat{\theta}$ through maximum likelihood estimation, which can be optimized for by interleaving stochastic gradient descent with Gibbs sampling steps. Lastly, we obtain our probabilistic training labels $p_{\hat{\theta}}(Y|\Lambda)$.

IV. METHODS

Our goal is to demonstrate the effect that modeling dependencies under various settings, regarding the number of LFs and dependencies as well as their type and strength, has on end-model performance, *at best*, i.e. by modeling those dependencies that, under a given setting, reflect the *true* structure of the LFs.

A. Synthetic dependencies and labeling functions

Using synthetically generated dependencies and labeling function outputs is attractive since we not only have full knowledge over the *true* structure but also control over the number of labeling functions as well as dependencies and their type and strength we want to simulate.

To do so, we provide, the factor graph from section III, the graph's structure (i.e. the number of LFs and factors to be modeled) and corresponding model parameters θ. We can

then sample synthetic values for the random variables to generate a synthetic label matrix Λ. In particular, we can also provide arbitrary true labels Y^*, treat them as evidence (i.e. not as a random variable) and, thus, only sample synthetic LF votes corresponding to a true label.

The advantage of this is that we can provide labels corresponding to real datasets, thereby essentially generate synthetic labeling functions for this dataset and, importantly, use the corresponding training data (not used for generation of Λ) to train a downstream classifier on probabilistic labels learned, exclusively, from the synthetic label matrix Λ and the structure we chose to provide.

In essence, we generate a synthetic label matrix by reversing the usual process of providing an observed label matrix to learn the model's parameters $\hat{\theta}$.

B. Finding true dependencies in real datasets

The underlying *true* structure of real labeling functions is, of course, unknown. Still, we can get fairly close to it by using true labels (together with the observed LF votes) to compute the resulting factor values for each data point, which we then sum up over the whole training set. For a given factor, we then choose to model the k dependencies with highest, positive total value. These are the dependencies for which the true labels provide the most evidence of being true.

For instance, with the LFs from Fig. 1, following the definition of a *fixing* dependency, we would add up all the instances where LF1 labels positively, LF2 negatively and the true sentiment is negative and then subtract all the instances where LF1 abstained but LF2 did not (none in this case). Semantically, the total factor value would be equivalent to the number of times where "wouldn't recommend" appears in a text sample and the corresponding sentiment is indeed negative. We stress that we exclusively use the true training labels for the purpose above.

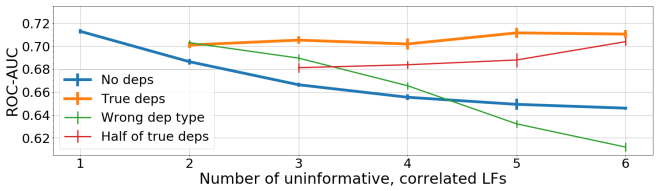
V. EXPERIMENTS

For the following experiments, we use the IMDB Movie Review Sentiment dataset[1], consisting of 25k training and test samples each [13].

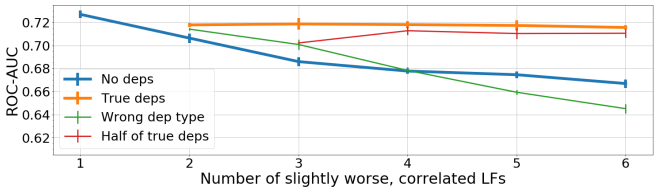
A. Correlated LF cluster

We first aim to find out how much we can lose up on downstream performance when ignoring the dependencies of a growing cluster of correlated and less accurate LFs.

Using the method described in IV-A we first generate a label matrix that corresponds to synthetic votes of 10 labeling functions on the training labels of the IMDB dataset (all LFs with coverage of 5%). Four of these labeling functions are conditionally independent as well as fairly accurate (around 80%). The rest is a cluster of correlated LFs (i.e. we use the *similar* factor) that are pretty uninformative (accuracy between 53 and 56%). We then start with the set of four accurate LFs to which we incrementally add one of the inaccurate ones and model i) No, ii) All, iii) Half of all, and iv) *bolstering* instead of *similar* dependencies. The plotted results are shown in Fig. 2a.



(a) Large accuracy difference



(b) Small accuracy difference, but stronger correlation

Fig. 2. (a), (b) We incrementally add correlated LFs to an initial set of four, fairly accurate, LFs. Modeling the correlatedness (the true *similar* factors) keeps initial performance, while ignoring them loses up to 6 AUC points on the right of (a).

As expected, modeling these *similar* dependencies impedes a performance drop, regardless of the number of correlated and inaccurate LFs we add. Even if we only model half of the *similar* dependencies, we can almost keep the initial performance, especially as the added LF cluster becomes large. On the other hand, ignoring the dependencies introduces their substantial noise to the factor graph and we can lose up to 6 AUC points. Finally, if we model the LF votes from within the cluster as *bolstering* each other (i.e. we wrongly encode into the model that when many of these LFs label equally, the true label likely is the same—green line), we, indeed, observe the largest drop in performance, though only as the number of uninformative LFs reaches 5.

In a similar experiment (Fig. 2b), we lower the difference in accuracy between initial set and cluster LFs, that is, the correlated LFs in the cluster are around 10% less accurate (i.e. close to 70%) than the initial ones. In addition, we let the cluster LFs be stronger correlated (with parameter weight, in log-scale, around 0.75 instead of the previous 0.25). We again observe a, slightly weaker, drop in performance when not modeling any (or wrong) dependencies. Interestingly, adding these strongly correlated but reasonably accurate LFs does not help in performance at all.

B. IMDB experiments

For the experiments in this section we simulate the usual data programming approach [1] [11] and manually select a set of 135 sensible, real labeling functions (some summary statistics are provided in Table. I). These LFs label on the presence of a single word or a pair of words (i.e. uni-/bigram LFs) such as the ones in Fig. 1. We deliberately choose the bigram LFs so as to create dependencies we expect to help with downstream model performance.

1) Effect of modeling dependencies on all 135 LFs: In this experiment we show i) the broad impact that modeling

[1]https://ai.stanford.edu/~amaas/data/sentiment/

TABLE I

SUMMARY STATISTICS OF THE 135 LFS WE SELECTED FOR THE IMDB DATASET REGARDING THEIR ACCURACY (ACC.) AND COVERAGE (COV.)

Mean LF acc.	LF acc. std	Mean LF cov.	LF cov. std
0.7787	0.1306	0.0153	0.0373

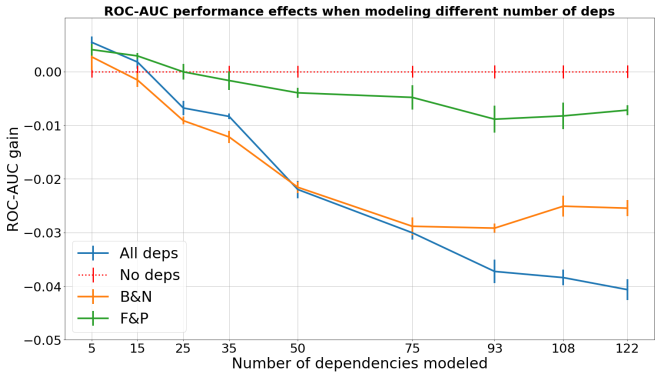


Fig. 3. Modeling more than a handful dependencies (as the ones in Table II) significantly deteriorates ROC-AUC downstream performance as compared to simply ignoring them (red line), which gives the baseline score of 0.8098. This effect intensifies as we model more dependencies.
B&N — We only model the *bolstering* and *negated* dependencies.
F&P — We only model the *fixing* and *priority* dependencies.

dependencies has on end-model performance for the set of real LFs detailed above and ii) the changing impact as we model more and more dependencies.

We choose different $k \in \{1, 3, 5, \ldots, 40\}$ and then model the strongest, as per IV-B, $\leq k$ dependencies of each factor (we exclude the *similar* one in this experiment, and model $< k$ dependencies only when there are no more dependencies with positive total factor value). A selection of the strongest and weakest dependencies is shown in Table II. Note that the weakest ones still make sense and only are modeled in the case where we take the ≤ 40 strongest dependencies.

While we observe a marginal boost (< 0.005) in performance when modeling the strongest $k = 1, 3$ dependencies of each factor (i.e. $5, 15$ in total), the main take-away is the following:

We surprisingly find that modeling more than a handful dependencies significantly *deteriorates* the downstream performance (by up to 4 ROC-AUC points) as compared to simply ignoring them (Fig. 3). Moreover, the performance worsening intensifies as we increase k, i.e. as we model more, slightly weaker, dependencies. We reiterate that these additional dependencies still, semantically, make sense (as depicted in Table II, where the weakest ones are modeled for the case where $k = 40$ / number of dependencies $= 122$), and indeed have positive factor values, which indicates that they provide positive evidence for the true labels we aim to estimate. Even though this result and insight is highly relevant for practitioners, it has, to the best of our knowledge, not been known or researched. It comes as a surprise that modeling seemingly sensible dependencies can significantly deteriorate the targeted downstream model performance.

TABLE II

THE STRONGEST AND WEAKEST DEPENDENCIES FOR THE IMDB LFS

LF_j	LF_k	factor value	factor
best	great	801	bolstering
bad	don't waste	110	bolstering
worth	not worth	238	fixing
great	nothing great	15	fixing
worth	not worth	219	negated
special	not special	8	negated
original	bad	327	priority
recommend	terrible	53	priority
recommend	highly recommend	226	reinforcing
bad	absolutely horrible	7	reinforcing

2) Repeatedly subsampling from the 135 LFs: In this set of experiments, we repeatedly subsample 10 to 70 LFs out of the 135 we selected above. For each such subset, we again model different numbers of dependencies as before.

While for some subsets we observe (Fig. 4) lifts as well as drops in performance (of at most 3% each), most often the impact of modeling dependencies is only *marginal*. We identify the baseline performance of not modeling dependencies as a factor for the observed lifts. That is, when the labeling functions alone do not provide enough evidence for strong downstream model performance, the addition of modeling dependencies may potentially help. Conversely, we find that modeling dependencies can only provide *marginal* additional evidence for estimating the true label when the signal coming from the LFs alone is already strong — and, in some of these cases, as in the experiment above with all 135 LFs, it even *worsens* performance. Furthermore, we find that *no deteriorating* behavior happens for subsets of cardinality ≤ 30.

C. Implementation details

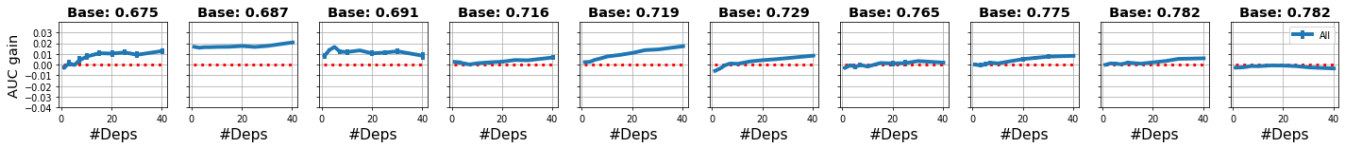
For all reported experiments we average over 5 runs for the generative model, where for each such run we use the estimated probabilistic labels to train a multilayer perceptron (MLP) 20 times, which we again average out. That is, we report the mean performances on the test set, averaged over 100 runs.

The MLP, the downstream model, has two hidden layers of size 20 with ReLU activations, sigmoid output and cross entropy loss. We train it for 250 epochs using the Adam optimizer.

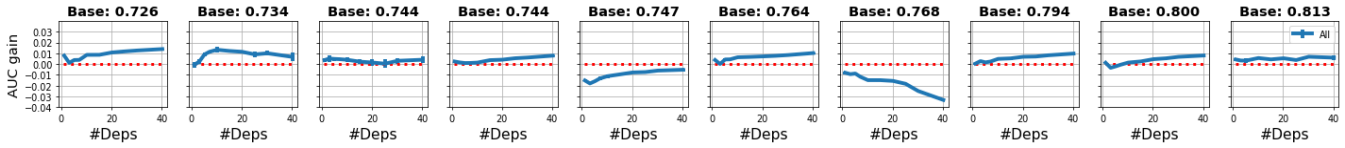
As training data we use low dimensional projections of a bag-of-words matrix via truncated Singular Value Decomposition (SVD), fixing the embedding size to d = 300 — with the goal of not adding, for our purpose, unnecessary complexity through, e.g., pre-trained embeddings.

VI. CONCLUSIONS

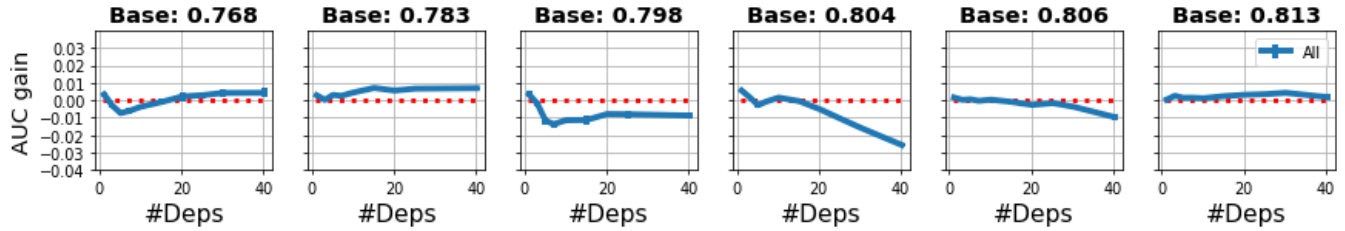
We have demonstrated various, significant impacts that modeling dependencies can have on downstream performance, i.e. *boosting*, *negligible* or even *deteriorating* effects. This may come surprisingly and emphasizes the importance of our presented work to gain, and give practitioners, a better



(a) Number of labeling functions: 20



(b) Number of labeling functions: 40



(c) Number of labeling functions: 70

Fig. 4. We repeatedly subsample 10 to 70 LFs out of the 135 we selected for the IMDB dataset and plot the gain in performance when modeling more and more, slightly weaker, dependencies (w.r.t. **Base**, the baseline AUC performance for not modeling any dependencies of a given subset—red dotted line).

understanding of the additional step of modeling dependencies. In particular, we have empirically shown that, in many settings of practical relevance, modeling seemingly sensible dependencies between weak sources of supervision has marginal or even negative impacts on the test performance of the downstream classifier. As such, we conclude that ignoring dependencies is a reasonable baseline for practitioners, as the, often small, lifts are confronted with potentially worse performances.

In future work, we plan to utilize this information to make well-founded decisions on when to interactively learn dependencies between weak supervision sources.

APPENDIX

A. Factor Definitions

We supplement the factor definitions from section III. For the *fixing* dependency we have:

$$\phi^{Fix}_{i,j,k,1} = \mathbb{1}\{\Lambda_{i,j} = -y_i^* \wedge \Lambda_{i,k} = y_i^*\}$$
$$\phi^{Fix}_{i,j,k,2} = -\mathbb{1}\{\Lambda_{i,j} = 0 \wedge \Lambda_{i,k} \neq 0\},$$

for the *negated*:

$$\phi^{Neg}_{i,j,k,1} = \mathbb{1}\{\Lambda_{i,j} = -y_i^* \wedge \Lambda_{i,k} = y_i^*\}$$
$$\phi^{Neg}_{i,j,k,2} = -\mathbb{1}\{(\Lambda_{i,j} = y_i^* \wedge \Lambda_{i,k} = -y_i^*) \vee \Lambda_{i,j} = \Lambda_{i,k} \neq 0\},$$

for the *bolstering*:

$$\phi^{Bol}_{i,j,k,1} = \mathbb{1}\{\Lambda_{i,j} = \Lambda_{i,k} = y_i^*\}$$
$$\phi^{Bol}_{i,j,k,2} = -\mathbb{1}\{\Lambda_{i,j} = -\Lambda_{i,k} \neq y_i^* \vee \Lambda_{i,j} = \Lambda_{i,k} \neq y_i^*\},$$

for the *similar*:

$$\phi^{Sim}_{i,j,k} = \mathbb{1}\{\Lambda_{i,j} = \Lambda_{i,k}\}$$

and, finally, for the *reinforcing* one:

$$\phi^{Rei}_{i,j,k,1} = \mathbb{1}\{\Lambda_{i,j} = \Lambda_{i,k} = y_i^*\}$$
$$\phi^{Rei}_{i,j,k,2} = -\mathbb{1}\{\Lambda_{i,j} = 0 \wedge \Lambda_{i,k} \neq 0\}.$$

ACKNOWLEDGMENT

This work was made possible through CMU's Robotics Institute Summer Scholars (RISS) program and the German Academic Exchange Service (DAAD) and very much supported by the members of the Auton Lab — special thanks going to Ben — as well as the RISS directors Dr. John Dolan and Ms. Rachel Burcin.

REFERENCES

[1] A. Ratner, C. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," *Advances in neural information processing systems*, vol. 29, 05 2016.

[2] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, "Training complex models with multi-task weak supervision," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4763–4771, 07 2019.

[3] D. Y. Fu, M. F. Chen, F. Sala, S. Hooper, K. Fatahalian, and C. Ré, "Fast and three-rious: Speeding up weak supervision with triplet methods," *ICML*, 2020.

[4] S. Bach, B. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," 03 2017.

[5] P. Varma, F. Sala, A. He, A. Ratner, and C. Ré, "Learning dependency structures for weak supervision models," *ICML*, 2019.

[6] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. Rubin, and C. Ré, "Inferring generative model structure with static analysis," in *Advances in neural information processing systems*, 2017, pp. 240–250.

[7] [Online]. Available: https://github.com/snorkel-team/snorkel

[8] S. H. Bach, D. Rodriguez, Y. Liu, C. Luo, H. Shao, C. Xia, S. Sen, A. Ratner, B. Hancock, H. Alborzi, R. Kuchhal, C. Ré, and R. Malkin, "Snorkel drybell: A case study in deploying weak supervision at industrial scale." New York, NY, USA: Association for Computing Machinery, 2019.

[9] P. Varma and C. Ré, "Snuba: Automating weak supervision to label training data," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 12, no. 3. NIH Public Access, 2018, p. 223.

[10] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *Journal of Machine Learning Research*, vol. 15, pp. 2773–2832, 2014.

[11] A. Ratner, S. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: rapid training data creation with weak supervision," *The VLDB Journal*, vol. 29, 07 2019.

[12] P. ling Loh and M. J. Wainwright, "Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2087–2095.

[13] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150.

[14] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 20–28, 1979.

Path Optimization for Autonomous Rough Terrain Traversal

Gargi Sadalgekar[1], Sean J. Wang[2] and Aaron M. Johnson[2]

***Abstract*— Robots capable of traversing rough terrain have many useful and varied applications. Many of these require the robot to travel to a desired location using the most efficient path possible. However, because of the non-linear and non-deterministic dynamics that robots have when traversing rough terrain, the process of determining the optimal path becomes difficult. While the robot's non-deterministic dynamics allow it to be modeled as a Markov decision process (MDP) problem, the existing methods for solving such a problem are computationally expensive. In this paper, we propose a new method for solving the MDP problem that utilizes a sampling-based optimizer called the cross-entropy (CE) method. We implement the CE method using a multivariate Gaussian distribution and evaluate the sample paths using a cost function that calculates each sample's overall path length and distance from the target endpoint.**

***Index Terms*— Autonomous Vehicle Navigation, Motion and Path Planning, Probability and Statistical Methods, Wheeled Robots**

I. INTRODUCTION

Robots with the ability to traverse extremely rough terrain have a wide variety of uses, such as terrain exploration, environmental monitoring, and search-and-rescue missions. A key requirement of many of these applications is that the robot is capable of travelling to a designated point using the most efficient path possible. Specifically, we are looking at the case where a wheeled robot with high suspension and traction is tasked with traversing rough terrain in order to survey a region, as shown in Fig. 1.

If a robot were traveling over flat ground, the problem of finding the optimal path would be straightforward because the robot's flat-ground dynamics would classify it as a Dubins' car. A Dubin's vehicle has an established optimal path consisting of the concatenation of a straight line segment and two circular arcs of maximum curvature [1]. However, determining the optimal path for a robot to take when traversing rough terrain is more complicated. The established optimal path for a Dubin's car is not guaranteed to work when executed over rough terrain because it doesn't take into account the new complexities of the environment. If, for instance, there is uneven terrain along the established optimal path, the robot could get stuck or flip over, causing it to fail in reaching the desired point. In such a case, the true optimal path would navigate the robot around such obstacles while still reaching the desired endpoint in an efficient manner.

[1]Gargi Sadalgekar is with the Robotics Institute Summer Scholars Program at Carnegie Mellon University, Pittsburgh, PA 15213, USA and also with the Department of Mechanical and Aerospace Engineering at Princeton University, NJ 08540, USA gargis@princeton.edu

[2]S.J. Wang and A. M. Johnson are with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA {sjw2, amj1}@andrew.cmu.edu

Fig. 1. Four-wheeled steered robot with four-link off-road suspension traversing rough terrain.

Furthermore, even if there was some way to immediately identify the optimal path, traditional linear control systems would have difficulty keeping the robot on said path. This is because rough terrain traversal involves highly non-linear dynamics as the robot makes and breaks contact with obstacles in complex shapes. As a result, controllers based on linear dynamics offer poor results, and an alternative method for control is necessary. One such alternative is model predictive control (MPC), where a model is used first to predict the motion of a robot when given a sequence of actions and then to continually recompute the optimal action-sequence as the robot diverges from the predicted motion. This type of control requires a model of the system based on the robot's dynamics. Over rough terrain, these dynamics become non-deterministic due to unknown variables, such as friction, payload, or complex terramechanics, so the system can be modeled using a Markov decision process (MDP). The problem of finding an optimal path thus reduces to one of solving the MDP of the system. However, most traditional methods for solving MDP problems, such as value iteration and policy iteration, are inefficient. Both methods involve updating the value function of every state at every iteration, which becomes computationally expensive over time [2]. For MPC to be effective, it requires a fast method for solving the MDP problem in order to efficiently reach the goal.

In this paper, we propose implementing a sampling-based optimizer called the cross-entropy (CE) method as a faster means for solving the MDP problem and computing the optimal action-sequence to reach a designated goal. We use a data-driven probabilistic motion model of the robot's dynamics and input a sequence of actions that the model

uses to predict the robot's motion as it executes the actions. The optimal action-sequence that will take the robot to the desired goal is determined using the CE method. The CE method randomly samples action-sequences from a probability distribution and uses the action-sequences with the lowest associated costs to re-weight the distribution. This process is repeated until the distribution eventually converges over the optimal action-sequence. In contrast to the traditional approaches to solving an MDP problem, the proposed approach is able to solve the MDP problem associated with the non-deterministic dynamics of the system using a fast method for computation that can be done in real time.

II. Design

A. The Cross-Entropy Method

The cross-entropy method is an adaptive importance sampling procedure that is used for rare-event probability estimation and optimization. Here we provide a brief summary of the CE method. More details can be found in [3].

Consider the probability ℓ that a real value function $S(\mathbf{x})$ is below some threshold γ, where $\mathbf{x}$ is a random variable selected from probability distribution function (pdf) $f(\cdot;\mathbf{u})$ which is parameterized by a finite-dimensional real vector $\mathbf{u}$.

$$\ell = \mathbb{P}(S(\mathbf{x}) \leq \gamma) = \int I_{S(\mathbf{x})\leq\gamma} f(\mathbf{x};\mathbf{u}) d\mathbf{x} \tag{1}$$

Here the expression $I_{S(\mathbf{x})\leq\gamma}$ represents an indicator function that equals one when the expression $S(\mathbf{x}) \leq \gamma$ is true, and zero when the expression is false. The rare-event probability ℓ can be represented using another pdf g as follows:

$$\ell = \int \frac{I_{S(\mathbf{x})\leq\gamma} f(\mathbf{x};\mathbf{u})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x}. \tag{2}$$

Then, if we draw N random variables $\mathbf{x}^1$,...,$\mathbf{x}^N$ from g, we can define an unbiased importance sampling estimator of ℓ as $\hat{\ell}$ where

$$\hat{\ell} = \frac{1}{N} \sum_{k=1}^{N} I_{S(\mathbf{x}^k)\leq\gamma} \frac{f(\mathbf{x^k};\mathbf{u})}{g(\mathbf{x^k})}. \tag{3}$$

The optimal importance sampling pdf that minimizes the variance of $I_{S(\mathbf{x})\leq\gamma} \frac{f(\mathbf{x};\mathbf{u})}{g(\mathbf{x})}$ is defined as g^*, where

$$g^*(\mathbf{x}) = \frac{f(\mathbf{x};u) I_{S(\mathbf{x})\leq\gamma}}{\ell}. \tag{4}$$

The goal of the CE method is to choose an importance sampling pdf g, within the class of pdfs $f(\cdot;\mathbf{v})$, such that the Kullback-Leibler (KL) divergence between g and an optimal importance sampling pdf g^* is minimal. The minimization problem can be simplified to finding the optimal reference parameter $\mathbf{v}^*$.

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}} \frac{1}{N} \sum_{k=1}^{N} I_{S(\mathbf{x}^k)\leq\gamma} \frac{f(\mathbf{x}^k;\mathbf{u})}{f(\mathbf{x}^k;\mathbf{w})} \ln f(\mathbf{x^k};\mathbf{v}) \tag{5}$$

Since ℓ is a rare-event probability, $\mathbf{v}^*$ cannot be found in one step as most of the indicators $I_{S(\mathbf{x})\leq\gamma}$ in (5) will be zero, and the maximization problem will become moot. Instead, a multi-step CE procedure is used where a new reference parameter $\mathbf{v}_t$ and a new threshold γ_t are calculated at each step with the goal that the reference parameter converges to $\mathbf{v}^*$, once the threshold reaches γ. At each iteration t, we generate N samples from the current pdf, calculate the associated $S(\mathbf{x})$ values, and define an intermediary gamma value γ_t as the ρ-quantile of the performance values $S(\mathbf{x})$. The rarity parameter ρ is a user-specified parameter which ensures that, at each iteration, a set number $N^e = \lceil \rho N \rceil$ of the samples taken have an indicator of 1. This set of N^e variables for which $S(\mathbf{x}) \leq \gamma_t$ is called the elite set. Once γ_t is determined, the value of the reference parameter is updated using (5) based on the elite set. The iterations continue until γ_t reaches the desired threshold of γ. The probability ℓ can then be estimated using (3) where $\mathbf{u}$ equals the reference parameter found in the final iteration.

B. The CE Method for Optimization

This method for estimation can also be adapted for optimization purposes, since the probability of reaching some optimal solution using a random search is a rare-event probability. Thus, the CE method can be used to gradually change the sampling distribution of the random search so that the selection of the optimal solution is more likely to occur. In this case, the optimal solution refers to some variable $\mathbf{x}^*$ that either minimizes or maximizes a given function. For example, suppose that the goal is to find the value $\mathbf{x}^*$ that yields the minimum of a function $S(\mathbf{x})$. This unknown minimum can be denoted as follows

$$S(\mathbf{x}^*) = \gamma^*. \tag{6}$$

This can then be translated into the following estimation problem

$$\ell = P(S(\mathbf{x}) \leq \gamma^*) \tag{7}$$

to find the lowest possible γ^* through an iterative process.

Before initiating the iterations, an initial reference parameter $\mathbf{v}_0$ is defined along with a variable N^e that represents the size of the elite set. Each iteration t starts by sampling N random variables from the chosen pdf, written as $\mathbf{x}_t^1, ..., \mathbf{x}_t^N \sim_{iid} f(\cdot;\mathbf{v}_{t-1})$. Then the cost function $S(\mathbf{x})$ is evaluated at $\mathbf{x}_t^k$ for all k. An intermediate threshold γ_t is defined as the $(N^e+1)^{th}$ largest cost. Using this threshold, the reference parameter $\mathbf{v}_t$ is then updated with

$$\mathbf{v}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \frac{1}{N} \sum_{k=1}^{N} I_{S(\mathbf{x}_t^k)\leq\gamma_t} \ln f(\mathbf{x}_t^k;\mathbf{v}). \tag{8}$$

This process is repeated until some user-defined stopping parameter is met. Algorithm 1 provides a complete summary of the process.

To run this algorithm, the user also needs to select the appropriate probability distribution function family. If the chosen distribution belongs to an exponential family of sampling distributions, then the optimal reference parameter $\mathbf{v}_t$ of each iteration can be solved for explicitly. For example, the reference parameter of a Gaussian distribution can be

found by solving for the mean and variance of the elite set. Furthermore, the user needs to establish a stopping criterion for the iterations. One option is to stop iterating once the change in the distribution between iterations, measured by the KL-divergence, falls below some threshold ϵ. Similarly, the user can also stop iterating once the change in γ_t falls below ϵ. If the probability distribution being used is a Gaussian, the algorithm can also be terminated when the determinant of the Gaussian's covariance matrix approaches zero [4]. For the purposes of this paper, we use the change in γ_t as the stopping criterion.

Algorithm 1 CE Algorithm for Optimization

1: Choose an initial parameter vector $\mathbf{v}_0$
2: Define $N^e = \lceil \rho \cdot N \rceil$
3: Set t = 1
4: **while** $\Delta\gamma < \epsilon$ **do**
5: Generate $\mathbf{x}_t^1$,...,$\mathbf{x}_t^N \sim_{iid} f(\cdot; \mathbf{v}_{t-1})$
6: Calculate the performances $S(\mathbf{x}_t^k)$ for all k
7: Sort the samples such that $S^1 \leq ... \leq S^N$
8: Let $\gamma_t = S^{N^e+1}$
9: Update $\Delta\gamma = \gamma_t - \gamma_{t-1}$
10: Update $\mathbf{v}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \frac{1}{N} \sum_{k=1}^{N} I_{S(\mathbf{x}_t^k) \leq \gamma_t} \ln f(\mathbf{x}_t^k; \mathbf{v})$
11: $t = t + 1$
12: **end while**

C. Application of CE Method for Rough Terrain

In this section, we implement the CE method to find an action-sequence that accurately sends the robot to the desired goal using the least amount of steps over the shortest possible path. Each action in the action-sequence represents a pair of inputs that the robot receives: one value corresponds to the throttle input and the other value corresponds to the steering input. As a result, each sampled sequence $\mathbf{x}$ has the dimensions $2L$, where L represents the number of actions in a sequence. The performance of each sample is quantified using a cost function $S(\mathbf{x})$ that measures criteria such as path length, steps taken, and proximity to the target. Thus by minimizing the cost function the CE method is able to determine the optimal action-sequence.

The first step in finding the optimal path is to randomly select a set of action-sequences from a multivariate Gaussian distribution, where each variable sampled from the pdf is actually a vector, and each component of the vector represents one action in an action-sequence. Since the robot requires two inputs for each action command, two multivariate Gaussians are needed to draw from, one for each input. Once N variables are sampled from each distribution, the corresponding throttle and steering inputs are paired to create N action-sequences. The cost of each action-sequence is then evaluated using the cost function $S(\mathbf{x})$. This is done by first simulating each action-sequence through the environment using a probabilistic motion model and then evaluating it based on the resulting path. Once all of the costs are calculated, the next step is to determine which action-sequences belong in the elite set mentioned in section II-A. The threshold γ_t is defined using the $(N^e+1)^{th}$ largest cost and for every cost below γ_t, the corresponding throttle and steering variables are each assigned to the elite set for each distribution. Each elite set is then used to update the reference parameter, $\mathbf{v}_t$, for each distribution. As mentioned earlier, the updated reference parameter for a Gaussian distribution can be solved for explicitly by first finding the mean and covariance of each elite set as follows:

$$\hat{\mu}_t = \frac{1}{|\mathcal{E}|} \sum_{\mathbf{x} \in \mathcal{E}} \mathbf{x}, \quad \hat{\Sigma}_t = \frac{1}{|\mathcal{E}| - 1} \sum_{\mathbf{x} \in \mathcal{E}} (\mathbf{x} - \mu)(\mathbf{x} - \mu)^T \quad (9)$$

where $\mathcal{E}$ represents the elite set [4]. Once the mean and covariance of each elite set are determined, it is necessary to include a smoothing step that prevents the algorithm from converging to a sub-optimal solution [5]. Smoothing involves updating the mean and covariance such that:

$$\mu_{t+1} = \alpha * \mu_t + (1 - \alpha) * \hat{\mu}_t, \quad \Sigma_{t+1} = \alpha * \Sigma_t + (1 - \alpha) * \hat{\Sigma}_t \quad (10)$$

where α is a user-defined smoothing parameter. Once the mean and covariance are updated for each distribution, a new set of samples is drawn from each pdf and the entire process is repeated. The iterations continue until the change in γ_t falls below a designated threshold, indicating that the distributions have converged over a path.

For this procedure, the user needs to define a sample size N, the number of steps taken by the robot in each action-sequence L, the rarity parameter ρ, the smoothing parameter α, the stopping parameter threshold, an initial mean vector μ_0, and an initial covariance matrix Σ_0, where

$$\mu_0 \in \mathbb{R}^L, \quad and \quad \Sigma_0 \in \mathbb{R}^{L \times L} \quad (11)$$

D. Cost Functions

A large contributor to the success of the algorithm is the type of cost function used to evaluate each sampled path. We tested two different cost functions to see which one produced a more optimal path. The first cost function $S_1(\mathbf{x})$ evaluates each action-sequence based on the Euclidian distance from the desired endpoint to the last position of the robot once the action-sequence is completed. The second cost function $S_2(\mathbf{x})$ evaluates each action-sequence based its overall path length and the distance from the desired endpoint that is determined in $S_1(\mathbf{x})$. Function $S_2(\mathbf{x})$ calculates the path length by totalling the distances between every action in the action-sequence. This sum, denoted as $dist$, is then combined with the distance to the desired endpoint to produce the following cost function

$$S_2(\mathbf{x}) = dist \cdot \beta + S_1(\mathbf{x}) \quad (12)$$

The variable β is included as a user-defined weighting parameter that is included to allow the user to decided whether the algorithm should give preference to paths that are closer to the desired endpoint or to paths with a shorter path length. Another cost function that could potentially be effective is one that evaluates an action-sequence based on

the amount of time the robot spends at the endpoint. Such a cost function would identify the optimal path as one where the robot reaches the endpoint in fewest number of steps and then remains there for the remainder of the action-sequence.

III. Results

In order to determine which cost function produces the more optimal final path, we ran a series of trials to compare the performance of each cost function described in II-D. Table III outlines the values of all the user-defined parameters required to run the algorithm. Furthermore, all trials were given the coordinates (5.0, 5.0) as the desired endpoint for a robot starting at (0.0, 0.0). The trials were simulated using a flat-terrain motion model. The cost functions were compared based on the resulting optimal path's proximity to the target, its overall path length, and the number of iterations the algorithm needed to converge. Table III compares the average results of 30 trials.

TABLE I

A list of all the user-defined parameters used in the trials.

Parameter	Value	Description
N	300	Number of samples
L	75	Number of steps in path
ρ	0.99	Rarity Parameter
α	0.5	Smoothing Parameter
β	0.75	Weighting Parameter for $S_2(\mathbf{x})$
ϵ	0.0001	Stopping Parameter Threshold
μ_0	zero vector	Initial Mean Vector
Σ_0	identity matrix	Initial Covariance Matrix

TABLE II

Comparison of the optimal paths produced by each cost function based on path length and proximity to the target.

Cost Function	Final X Position (m)	Final Y Position (m)	Path Length (m)	Iterations
Ideal Path	5.000	5.000	7.071	-
$S_1(X)$	4.997	5.004	10.987	24.8
$S_2(X)$	4.998	5.000	8.669	35.8

According to the results of the trials, both cost functions are capable of bringing the robot within 5 mm of the desired endpoint. However, $S_2(\mathbf{x})$ produces action-sequences that reach the goal over a much shorter path length. Furthermore, Fig. 2 and Fig. 3 demonstrate the qualitative differences between the paths produced by the two cost functions, where the red plus in each indicates the desired endpoint for the path. Function $S_2(\mathbf{x})$ clearly produces optimal paths that are shorter and smoother than those produced by $S_1(\mathbf{x})$.

IV. Conclusion

The results of the trials demonstrate that the CE method for optimization can be used to determine a near-optimal action-sequence for a robot to reach a desired location. The adaptation of the CE method for this purpose involves sampling action-sequences from a multivariate Gaussian distribution and then quantifying the performance of each sample using a cost function that evaluates the action-sequence based on its path length and final distance from the desired endpoint. The least costly samples are used to gradually shift the Gaussian distribution towards an action-sequence with a minimal cost. The next section discusses what steps need to be taken in order to determine whether this method is indeed faster than existing methods for the MDP problem.

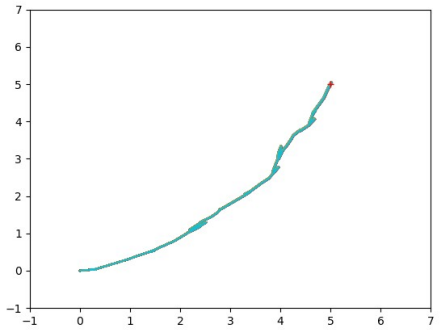


Fig. 2. Optimal path determined using $S_1(\mathbf{x})$

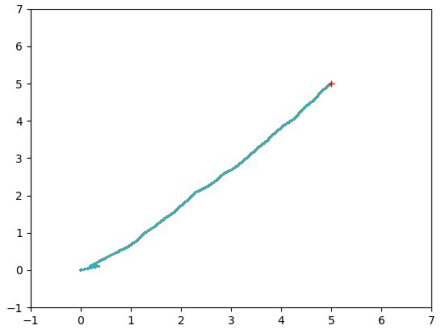


Fig. 3. Optimal path determined using $S_2(\mathbf{x})$

A. Future Works

At this point, the algorithm still requires modifications in order to consistently produce the most optimal action-sequence for a robot to arrive at a desired endpoint. A next step would be to develop a cost function that evaluates each action-sequence based on the amount of time, or the number of steps, that the robot spends at the desired endpoint. Such a cost function would ideally give preference to action-sequences that reach the target endpoint quickly and spend the remaining allotted steps at the target. Another important step would be to test alternative probability distributions, to see if there is a better way to represent the data. Specifically, we would like to test a version of the algorithm that uses a multivariate Gaussian mixture model based off of the algorithm proposed in [6]. Implementing a multivariate Gaussian mixture model would more accurately represent the data in scenarios where the data is multi-modal. Once these modifications are made, the next step would be to test the method using a rough-terrain motion model, especially since we have only been able to test the method with flat terrain models so far. The final step would be to compare this method's performance to that of other established methods in order to determine its validity.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. This work was also supported by the Carnegie Mellon Department of Mechanical Engineering. I would like to thank RISS for ensuring that I still had the opportunity to conduct robotics research even while working remotely, and I am especially thankful for all the hard work put in by Rachel Burcin and John Dolan to keep the RISS program running all summer. Next, I would like to thank my mentor Sean Wang, who devoted much of his time to help me through the RISS experience. I would also like to thank Dr. Aaron Johnson and the rest of the RoboMechanics Lab for welcoming me into the lab. Finally, I would like to thank my friends, Rei Zhang and Wilbur Wang, and my family for supporting me throughout this entire experience.

REFERENCES

[1] D. A. Anisi, "Optimal motion control of a ground vehicle," Ph.D. dissertation, Citeseer, 2003.

[2] P. Dai and J. Goldsmith, "Topological value iteration algorithm for markov decision processes." in *IJCAI*, 2007, pp. 1860–1865.

[3] Z. Botev, D. Kroese, R. Rubinstein, and P. L'Ecuyer, *Chapter 3. The Cross-Entropy Method for Optimization*, 12 2013, vol. 31, pp. 35–59.

[4] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012.

[5] Z. Botev and D. Kroese, "Global likelihood optimization via the cross-entropy method with an application to mixture models," 01 2005, pp. – 535.

[6] S. Geyer, I. Papaioannou, and D. Straub, "Cross entropy-based importance sampling using gaussian densities revisited," *Structural Safety*, vol. 76, pp. 15–27, 01 2019.

Online and Decentralized Multi-Agent Path Planning for Realistic Robotics Systems

Nayana Suvarna[1], Guillaume Sartoretti[2] and Howie Choset[3]

***Abstract*— Multi-agent path finding (MAPF) is the problem of determining collision-free paths for an arbitrary number of agents in a given workspace. Our group's prior work deals with scalability - being able to allow algorithms to operate with large numbers of agents - utilizing an algorithm called M*. Our next challenge, which is necessary for real-world deployments is to handle the uncertainties and conditions that real-world systems possess. Uncertainties caused by noise and imperfect localization and controls as well as large team sizes can cause costly replanning times with current state of the art approaches. To address these challenges, we propose a decentralized, online framework that distributes the work of a variant of M*, called ODrM*, to effectively plan for large sets of agents with minimal delay to their movement. We test our framework by placing increasing amounts of agents in a simulated environment and by comparing plan execution times as well as the ratio between "offline" pre-planning time using ODrM* to online plan execution time from our framework. Through our tests, we suggest that our framework will outperform ODrM* in plan execution time for all agent when scaled to larger team sizes.**

I. INTRODUCTION

Large scale robotics systems have recently been applied to various applications, including warehouses and industrial environments, as well as search and rescue. Most widely accepted algorithms in this field are centralized. Although these approaches can provide complete and optimal plans, they fail when applied to real-world deployments. Creating a single joint plan for all agents at the beginning of planning is infeasible due to position uncertainties caused by imperfect localization and controls. There is a growing need for algorithms that can scale well and handle these uncertainties.

Previous work with our group deals with a centralized, dynamically coupled algorithm called ODrM*. Although ODrM* outperforms other state of the art coupled planners with larger team sizes, planning time increases significantly when scaling to larger teams. Planning can take up to five minutes for bounded sub-optimal solutions for teams of over 150 agents [1]. This makes it impractical when applied to realistic systems because of the increased delay before agents can start moving to their goals.

This paper presents an online, decentralized path planning framework for multi-agent systems that uses ODrM* as the local planner to resolve collisions. Agents initially plan their individual paths assuming they are alone in the world. As agents proceed to their goals, they share plans with other agents that get within range. Plans are then compared and if a collision exists, all agents involved start replanning for the local subset of agents. The new plans are shared by the first agent who finishes planning and all agents can then resume their journey to their goals. If no collision exists, all agents just proceed to their goals and retain knowledge of plans shared by other agents, which they can use in the future as constraints for subsequent planning instances.

Through this approach, we leverage the benefits of decentralized, online planning to distribute the execution of ODrM* across all agents during movement. By only using ODrM* as the local planner when a collision is detected, we keep team sizes small which allows for faster replanning without delaying the movement of agents. We present results from tests where we place increasing numbers of agents in a simulated environment and demonstrate that the plan execution time for our framework is faster and the ratio of "offline" pre-planning" time to our framework's plan execution time increases as the number of agents increases.

II. PREVIOUS WORK

A. Multi-Agent Path Finding (MAPF)

Recent efforts for solving MAPF problems have revolved around decoupled and dynamically-coupled approaches. Decoupled approaches compute plans for agents individually and then resolve collisions by coordinating movement amongst them to avoid collisions. Optimal reciprocal collision avoidance (ORCA) [2] presents a velocity-based approach that infers agents' velocities while moving and distributes work amongst agents in adjusting individual velocities to avoid collisions. However, an assumption is made that all agents can perfectly assess the positions and velocities of other agents, and there is no communication between agents. Priority-based planners organize agents according to priority and plan paths for agents in decreasing order. Higher priority agents are treated as moving obstacles [3]. If resources are mostly spent on higher priority agents, collisions may occur before plans are calculated for lower priority agents. Overall, decoupled approaches can compute plans for agents very quickly, but they are not guaranteed to find solutions to all problems even if a solution exists.

Dynamically-coupled approaches are a hybrid of coupled planners, such as A*, that search in the joint configuration space of all agents and decoupled approaches to plan for multiple agents efficiently. They combine the benefits of better plan quality from coupled approaches and the faster planning time of decoupled approaches to efficiently plan for

[1]Nayana Suvarna is with the Computer Engineering Department at the University of Pittsburgh, Pittsburgh, PA 15213 nls71@pitt.edu

[2]Guillaume Sartoretti is with the Mechanical Engineering Department at the National University of Singapore, Singapore 119077 guillaume.sartoretti@nus.edu.sg

[3]Howie Choset is with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213 choset@andrew.cmu.edu

multiple agents. Conflict Based Search (CBS) [4] applies a series of constraints to the higher dimensional search space of a system so that collisions are resolved on lower-dimensional search spaces.

B. M* and Subdimensional Expansion

Our previous work [1][5] deals with a dynamically coupled algorithm called M*. This algorithm is a centralized, dynamically coupled approach that effectively utilizes subdimensional expansion to plan for large sets of agents.

Subdimensional expansion is a framework that plans for each agent individually, assuming they are alone in the world, and then locally expands the search space if a collision is detected. After "backpropagating" the paths of agents involved in a collision, a coupled planning approach is then used to explore alternative paths in the locally expanded search space, and collision-free paths are computed for all agents involved. [5]

M* is an implementation of subdimensional expansion in which the search space is represented as a graph. A* is used as the underlying planner to ensure that complete and optimal paths are found. [5] The main difference between M* and A* is that M* planning is limited to locally expanded search space when a collision occurs, instead of the joint configuration space for the whole system.

III. APPROACH

A. Local Planner

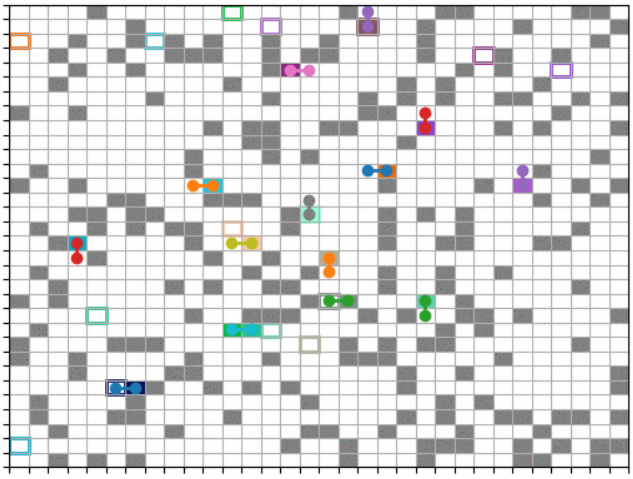


Fig. 1. Visualization of agents planning in a naive SE(2) workspace. The filled in boxes represent each agents current (x,y) position. The outlined boxes represent the goal for each agent with respect to the agents' fill color. The line indicates possible headings of 0°, 90°, 180°, and 270 °

B. Workspace Representation

With M* and it is variants, the configuration space that planning occurs on is R^2. The configuration space or workspace for a robot is the set of all possible configurations in a given environment. With an R^2 workspace, the robot's motion is limited to two dimensions, x and y, and the robot is abstracted away as a point on the Euclidean plane. With this configuration space, the robot can translate on the plane. [6] Although this works well when the motion of agents is limited to a theory-based graph environment, an additional factor of orientation needs to be considered when working with physical systems.

For our framework, we utilize a naive Special Euclidean (SE) group representation of the two-dimensional workspace (SE(2)) [6] to include orientation while planning. With this representation, agents now have to rotate in place to the appropriate orientation before translating in the plane. To simplify the problem, we planned in a grid-based environment. We assumed that each agent could only occupy one "cell" at any given timestep, and rotations are done in place. We limited the orientations to four possibilities: 0°, 90°, 180°, 270°. Our representation is classified as naive since only the x and y positions of agents are considered when checking for collisions. A "collision" is defined as when the two agents occupy the same x and y position at the same expected timestep. A visualization of agents planning on a naive SE(2) workspace using ODrM* is shown in Figure 1.

C. Framework

All agents initially plan to their goal, assuming they are alone in the world and proceed towards their goals. When agents get within a specific threshold distance, all agents in range share plans with each other. Upon sharing plans, every agent compares it's own path to every other agent's expected path to determine if a collision exists.

If a collision exists, all agents start replanning for the collision set using ODrM*. Whichever agent finishes planning first shares the new plans with all other agents involved. Upon receiving plans, agents stop their individual planning and start proceeding along their new paths. Agents then repeat this process until they reach their goals. Through our approach, we trade off suboptimality for time efficiency when scaling to larger agents. To further illustrate this concept, a figure that walks through an example situation with two agents is shown in Figure 2.

In the first panel, we see that both agents plan their individual paths to their goals. In the second panel, we see that the orange agent has rotated in place, and the blue agent has moved forward. At this point, both agents are within the threshold distance of each other. In panel 2, we can see that at this point, both agents share their plans. The blue agent retains information about the expected path for the orange agent, and the orange agent now retains information about the expected path for the blue agent. Both agents now know their own path as well as the other agent's path.

They both compare their individual paths to the other and determine that an imminent collision exists. Both agents start planning for themselves and the other agent using ODrM*. The blue agent finishes planning and shares the new plans with the orange agent. Upon receiving the plans, the orange agent stops it's individual planning and retains it's new plans and the new expected plans for the blue agent. This is demonstrated in the 3rd panel. The agents then proceed in the environment on their new paths and repeat this process

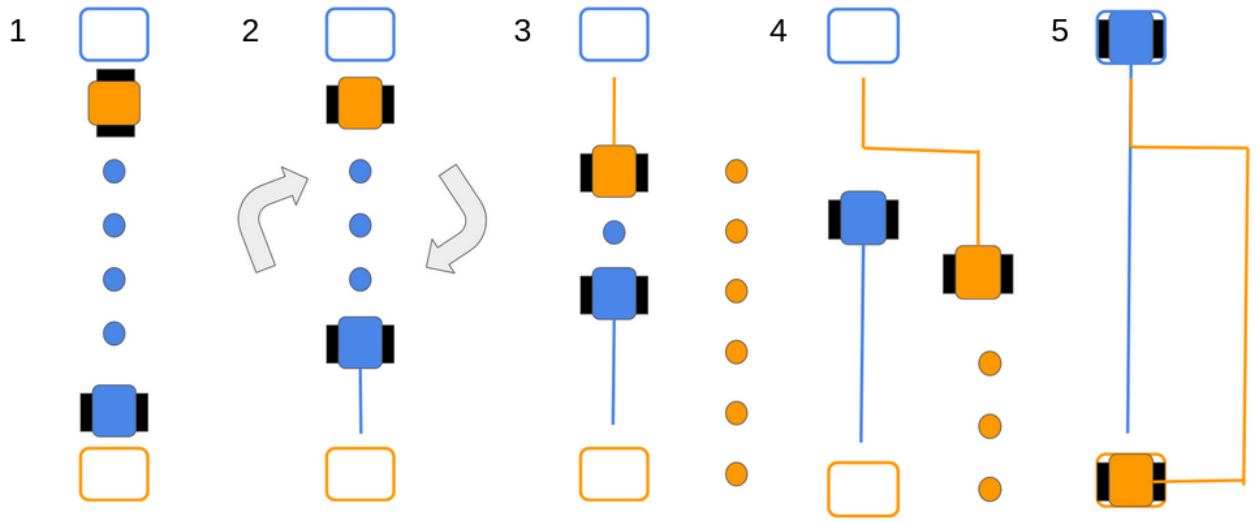


Fig. 2. Example of framework when applied to two agents. Filled in boxes represent the actual positions of agents. Outlined boxes represent goals. Smaller circles represent expected paths for agents and the solid lines represent the actual paths for agents.

until they reach their respective goals. This is shown in the last two panels.

D. Implementation

A critical aspect of our framework is that planning occurs in real-time during movement.To facilitate this, we implemented our framework using the Robot Operating System (ROS) and tested in a simulated Gazebo and Rviz environment. We utilized Turtlebot3 robots as our simulated agents and used the fake localization and move base packages to facilitate localization and local planning for each agent. We then used our framework as the global planner for each agent. This setup allowed us the overhead to plan while moving as well as to mimic the conditions and uncertainties that physical robotics systems provide.

IV. RESULTS

A. Comparison with ODrM*

For our experiments, we ran tests for sets of 4 to 16 agents. We set the distance threshold for all agents to 5 meters and limited the rate of planning to 5 Hz. Planning with ODrM* was uninflated. We placed agents in an empty world with no static obstacles and limited the map to an 18 meter by 18 meter area.

To compare our framework to ODrM*, we had to determine a set of metrics that could accurately compare the two approaches. The main difference we had to address was determining a relationship between path length (in agent moves) from ODrM* and simulation time from our framework. We compare OdrM* to our framework by using the total execution time of both approaches. The execution time was the amount of time that elapsed between the start of planning and when all agents reach their goals.

The total execution time for ODrM* was the sum of pre-planning and movement time. The pre-planning time was the amount of time that it took for ODrM* to compute

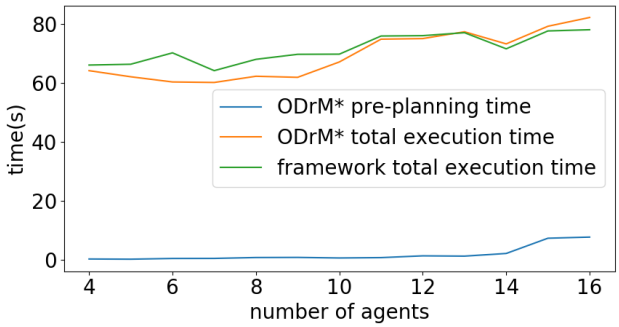


Fig. 3. Comparison between pre-planning time for ODrM*, total execution time for ODrM*, and total execution time for our framework

plans for all agents. The movement time was the expected amount of time for all agents to reach their goals using these plans. Since ODrM* returns paths in terms of number of agent moves, we compute the movement time for ODrM* by assuming that agents spend 3 seconds per agent move. Thus, the total execution time for ODrM* was the sum of pre-planning and movement time. For our framework, the movement time is equivalent to the execution time.

We found that our framework's total execution time was larger than the execution time for ODrM* when applied to less than 10 agents. Then, there is a balancing point from 10 to 13 agents where there is minimal error between the two approaches. The execution time for ODrM* exceeds our framework's execution time when applied to larger than 13 agents. Overall, as the number of agents increases, the total execution time for ODrM* surpasses the total execution time for our framework. This is demonstrated in Figure 3.

We also measured the "ratio of delay" with ODrM*. This was the ratio between pre-planning time for ODrM* and total execution time for our framework. As the number of

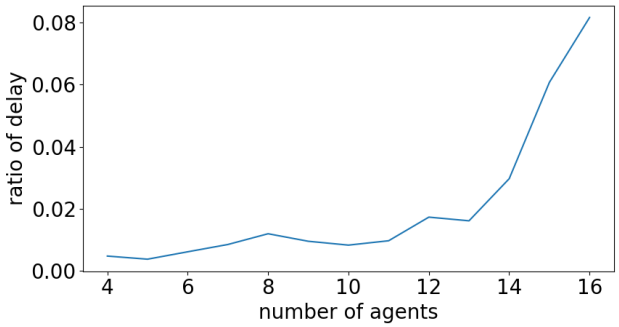


Fig. 4. Ratio of delay for ODrM*

agents increased, we saw an upwards trend in this ratio. This is demonstrated in Figure 4. Through this metric, we were able to quantify the amount of delay in the movement before agents can start moving to their goals when OdrM* is used.

Although more testing needs to be done, we are confident that these trends will carry over when applied to larger sets of agents.

V. CONCLUSION

In this paper, we presented a framework that is able to plan for agents in parallel to movement. We show through tests that we were able to successfully distribute the execution of ODrM* to run in real-time on agents with minimal delay in the execution of plans.

Future work will focus on applying our framework to larger sets of agents as well as maps with different densities to see how our metric compares with ODrM* and other similar MAPF approaches. We also believe that making the workspace representation for the underlying planner less naive may help overall performance. Further verification of our framework can also be conducted by applying it to physical robots.

ACKNOWLEDGMENT

Thanks to the Robotics Institute Summer Scholars Program as well as Rachel Burcin and John Dolan for facilitating the program this summer. Special thanks as well to Joshua Spisak and Andrew Saba for all their assistance over the course of the summer.

REFERENCES

[1] C. Ferner, G. Wagner, H. Choset, "ODrM*: optimal multirobot path planning for low dimensional search spaces."in *ICRA*, 2013, pp 3854-3859

[2] Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, 2011, pp 3-19

[3] M.Cap, P Novak M. Selecky,J.Faigl,and J.Voıneke ,"Asynchronous decentralized prioritized planning for coordination in multi-robot system," in *Proceedings of IROS*, 2013, pp 3822-3829

[4] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Conflict-Based Search For Optimal Multi-Agent Path Finding" in *Proceedings of AIII*, 2012.

[5] G. Wagner and H. Choset "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol 219, pp 1-24, 2015

[6] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, 2005, MIT Press.

Simulating Social Robot Quori using ROS and Gazebo

Adrian Thinnyun[1], Roshni Kaushik[2] and Reid Simmons[2]

***Abstract*— Quori is a novel and affordable socially-interactive robot platform designed for use in human-robot interaction (HRI) research. With an expressive projected face, two gesturing arms, and a bowing waist, Quori has the potential to easily generate versatile social behaviors. However, designing and implementing social skills and behaviors for Quori is currently a difficult task, due to the limited number of models available and the lack of a proper simulation environment. Having a proper simulation environment would make the development and testing of algorithms for Quori much easier. In this paper, we present our work towards a Quori simulator using ROS and Gazebo. We also showcase implementations of several behaviors for Quori that will be used in future work to determine how Quori's nonverbal movements relate to its perceived emotional state.**

***Index Terms*— simulation, animation, human-robot interaction**

I. INTRODUCTION

Socially-interactive robots have experienced a surge in popularity in recent years. While traditionally robots have been most popular in industrial settings such as automobile manufacturing, recent developments in human-robot interaction (HRI) have led to a shift from industrial robotics to service robotics [1]. Service robots often share the same physical space with people and regularly interact with them in both professional settings such as storefronts and research labs and domestic applications such as assisting the elderly. In order for robots to perform well in these applications, they must have strong capabilities for social interaction.

Two popular socially-interactive robots are Softbank's Pepper and Nao. Pepper and Nao are both humanoid robots that have been used in a wide variety of professional, research, and recreational activities. Pepper can be found in several Softbank stores around Japan waiting to greet customers and has proven to be equally popular with adults and children alike [2]. Nao has become a popular contender in the annual RoboCup league, which features intelligent autonomous robots competing in soccer [3]. Although both robots are widely used in labs and universities across the world, some research groups may not be able to afford the just-under $2000 price tag of Pepper [4], or Nao's nearly $8000 price [5]. Thus, having an affordable alternative may increase the opportunities for further HRI research.

Quori was designed to fulfill this role, released in the summer of 2018 with ten platforms awarded to ten research groups [6]. It was developed with input from the HRI community to prioritize the most important hardware capabilities for social interaction and maximize functionality while keeping costs affordable. For example, Quori's head consists of a low-cost projector allowing it to display various faces/expressions easily, and its arms are connected by 2-DOF shoulder joints that allow for simple gestures. A 1-DOF spine attaches this upper body to a holonomic mobile base, and options are available for mounting cameras, microphones, and other sensors to extend Quori's capabilities. Fig. 1 illustrates a sample configuration of Quori, though note that the modularity of its design allows for other configurations.

We use the open-source software Gazebo [8] as the environment for our simulator. Gazebo is a popular robot simulator that has already been used to simulate robots frequently used in HRI research, including both Pepper and Nao. A Unified Robot Description Format (URDF) [9] file is used to describe the virtual model of Quori, containing information about the various links of the model and their associated masses, inertia matrices, and joints. This information is extracted by the engine and used to render the robot in a 3D environment, as well as to simulate physical forces and enforce collision checking. We also use the ROS [10] framework in order to interact with the virtual robot model, sending actions for the robot to perform by publishing messages to the appropriate control topics.

In addition to developing a simulator for Quori, we also present work on developing non-verbal behaviors appropriate for social interaction. Humans use a wide variety of poses and gestures to convey emotions and other affective infor-

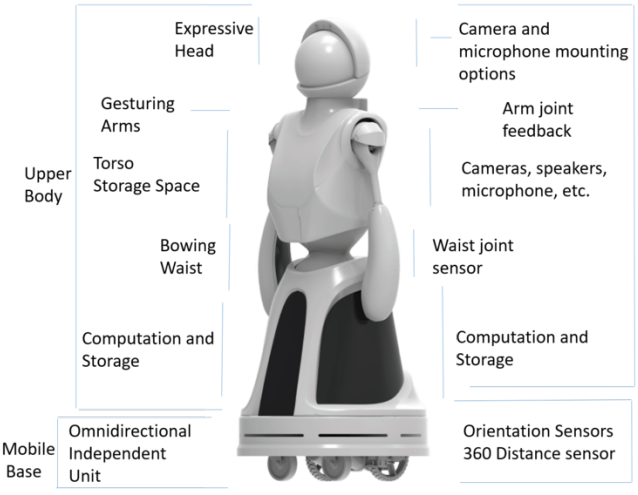


Fig. 1. Diagram of Quori illustrating its components and available degrees of freedom [7]

[1]Adrian Thinnyun is with the Department of Computer Science, University of Virginia, Charlottesville, USA athinnyun@gmail.com

[2]Roshni Kaushik and Reid Simmons are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, USA {roshnika, rsimmons}@andrew.cmu.edu

mation over the course of natural conversation. For example, someone experiencing anxiety or panic may lean backwards, while someone experiencing joy may move their arms up and down [11]. In turn, seeing someone express a certain emotion non-verbally often provokes an emotional response in oneself. Behaviors associated with disgust often create responses of fear, while behaviors associated with anger provoke more anger [12]. Thus, having the ability to produce these gestures seamlessly and accurately within the context of a conversation is critical to Quori's capability for social interaction.

One application of Quori's social behaviors we are particularly interested in is education. Recently there has been a promising body of work showing the effectiveness of the "learning by teaching" paradigm involving students instructing teachable robots [13]–[16]. Tanaka et al. used Pepper to teach children English through a variety of programs. In one activity, Pepper uses its tablet to display a lesson teaching the word "plane" and then extends its arms outwards to imitate the wings of a plane, inviting the children to join as they learn the word. In another activity, Pepper shows a video of a teacher teaching the word "mouth" while placing Pepper's hand on its mouth, then tells the children, "Teach me like that teacher is doing". This method, known as "direct teaching", has been shown to be effective in promoting children's learning [17].

Like Pepper, Nao has also seen use in experiments focused on improving education. Lemaignan et al. used Nao to help young children with motor difficulties improve their handwriting. Rather than Nao teaching the children how to write, the children were tasked with teaching Nao how to write, with the help of an occupational therapist. Although these children experience great deals of anxiety dealing with their handwriting difficulties, this scenario transforms the role of the child from a "bad writer" to a "teacher" helping Nao with its own handwriting, a strategy that greatly improves the child's self-esteem and motivation to participate in the activity. These studies demonstrate the potential of robots and the "learning by teaching" paradigm.

Quori could be a potential candidate for research of this nature, but before it can be used in any experiments it needs to be able to socially interact with humans by understanding the emotional state of its partner and responding with appropriate, context-driven actions. Previous work has shown a strong relationship between a student's nonverbal behaviors and their teacher's expectations and self-assessments [18], so care must be taken in order to make sure that Quori responds appropriately to a student mentor's instruction. In this paper, we present our progress on a Quori simulator and work towards developing a diverse set of nonverbal social behaviors.

II. SIMULATION OVERVIEW

In this section, we give a brief overview of the core components of our Quori simulation and present our original contributions to it.

A. Universal Robot Description Format (URDF)

At the heart of the simulation lies Quori's URDF file, which contains all of the necessary information for recreating Quori in a virtual environment. The most basic URDF file simply contains all of the links and joints comprising the robot defined in XML, where links are visually represented using either basic geometric shapes or pre-defined mesh files. However, in order to properly simulate a robot in Gazebo's physics engine, one must also add inertial information, including the mass and moment of inertia. Additional tags specifying colors and textures for each link, while not required, are helpful for accurately recreating the robot's true appearance and we have included this information in our URDF file. Finally, we include transmission tags describing the actuators of Quori's movable joints, as these allow us to manipulate our simulated robot using `ros_control`.

B. Robot Control in ROS

The `ros_control` [19] framework allows for control of the robot using standard ROS topics, nodes, and messages. A variety of controllers are available, each with its own parameters, message type, and function. We chose to use a `joint_trajectory_controller` as the main controller for Quori's joints. This controller accepts `JointTrajectory` messages as input, which contain a list of waypoints specifying the position, velocity, and acceleration of each joint at various time steps. We determined that this controller type provided the necessary freedom and flexibility to accurately produce the diverse set of movements needed for nonverbal communication.

C. Inverse Kinematics

To make the description of arm movements easier, we have also implemented an inverse kinematics solution that converts the 3-D desired position of the arm's end-point into the two shoulder angles required to reach the desired point. The equations for this solution were provided to us by the developers of Quori at the University of Pennsylvania.

For a given arm, we define θ_1 as the rotation of the arm's shoulder joint and θ_2 as its abduction/adduction (i.e. the raising or lowering of the arm). As depicted in Fig. 2, C_0 indicates the origin of the arm joint and p_a denotes an arbitrary point on the arm. Finally, we define the coordinate plane such that the x-axis points out to the front of the robot, the y-axis points out to the side of the robot from which the arm originates, and the z-axis is normal to the ground plane (assuming the torso is vertical). We chose to define the coordinate planes for the left and right arms as reflections of one another so that inputting the same command for both arms would lead to symmetrical movement. Thus, for an end-point of the arm $p_a = (x, y, z)$, the equations for the two joint angles are:

$$\theta_1 = \text{atan2}(z/x) \tag{1}$$

$$\theta_2 = \text{atan2}\left(\frac{\sqrt{x^2 + z^2}}{-y}\right) \tag{2}$$

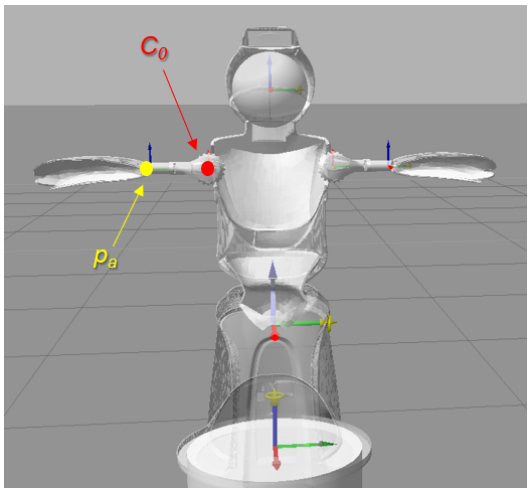


Fig. 2. Visualization of Quori, where C_0 indicates the origin of the arm joint and p_a denotes an arbitrary point on the arm (in this case, the origin of the lower arm)

This solution is implemented by a ROS node which subscribes to a topic consisting of `Float64MultiArray` messages specifying the arm(s) to be moved, the desired time for the action, and the Cartesian coordinates to be reached. The node converts these messages to `JointTrajectory` messages using Equations (1) and (2) and publishes them to the `joint_trajectory_controller`'s command topic.

III. NONVERBAL BEHAVIOR DESIGN

In this section, we discuss the key considerations and concepts involved in designing nonverbal social behaviors for Quori.

A. Facial and Bodily Expressions

With the necessary mechanisms in place for executing complex movements with Quori in a simulated environment, we focus our discussion on the human nonverbal behaviors we wish to emulate. In particular, we focus on bodily expressions of emotion rather than facial expressions. Originally it was believed that facial and vocal expressions alone were responsible for communicating the type of emotion being experienced while bodily movements/postures merely conveyed the intensity or level of arousal of the emotional state [20], [21]. However, this view has been challenged by recent studies showing that dynamic whole body and arm movement [22]–[25] as well as static body postures [26], [27] reveal specific information about a person's emotional state. Thus it is imperative to take Quori's whole body configuration into consideration when designing effective affective non-verbal behaviors.

B. The Action-Perception Disconnect

While the literature has found that certain body movements/poses seem to be associated with certain emotional states, performing those behaviors does not necessarily entail that an observer will perceive the correct emotional state from them. Perceived emotion may vary greatly depending on context, including factors such as the relationship between the actor and the observer, the environment in which the interaction takes place, the emotional state of the observer, etc. Thus, when designing nonverbal behaviors, we must acknowledge that we are merely optimizing behaviors towards the goal of successfully conveying a desired affect, rather than creating behaviors that do convey a desired affect.

With Quori, we encounter an additional disconnect: the disconnect between human and robot. Even if a certain human behavior always lead to a certain perceived emotional state and we were able to replicate it perfectly with Quori, we still would not be able to guarantee that an observer would interpret the behavior the same as they would if a human had performed it instead. Some people do not believe robots can have emotions, or that robots could take on the same roles as a human (e.g. student, teacher, salesperson, etc.). The visual differences between a robot and a human could also affect its perceived affect. Thus any behaviors we create for Quori will require further user validation before they can be deployed with confidence.

C. Quori's Limitations

Since Quori is a humanoid robot, it makes sense to use human body movements as a starting point when designing nonverbal behaviors for it, even if they may not be interpreted in the same manner. However, translating human body movements and gestures into motions Quori can perform is a difficult task. Consider the array of emotions that can be expressed by a single human hand. One can raise their fist to the sky, shake it back and forth, extend an open hand in front of them, or gently hold it above their head. Each of these actions could be used to make different inferences about the actor's intentions/emotions depending on the context and precise motions used. Quori lacks the ability to perform any of these actions as it has neither fingers nor a wrist. Similarly, simple gestures like nodding in agreement or shaking one's head in disagreement are unavailable due to Quori's lack of a movable neck joint.

Another limitation to consider is the speed at which Quori is able to move its joints. Humans often use quick, abrupt movements when expressing emotion, such as a sudden fist pump when feeling joy or quickly outstretching one's arms when feeling panic or fear. Quori's motors are only capable of producing relatively smooth, continuous motion, so these kinds of gestures are also unavailable to Quori. Quori's available degrees of freedom (DOF) are summarized in Table I.

TABLE I
QUORI'S DEGREES OF FREEDOM

Joint Name	Upper Limit	Lower Limit	Speed
Waist Hinge	0.47 radians	-0.27 radians	1 rad/sec
Turret Joint	N/A	N/A	3 rad/sec
Arm (Rotation)	N/A	N/A	1 rad/sec
Arm (Abduction)	1.1 radians	-1.1 radians	1 rad/sec

D. Mapping Body Movements to DOF

We now present our initial design choices for Quori's nonverbal behaviors. Table II features the six canonical emotions – happiness, sadness, fear, disgust, anger, and surprise – as well as the state of "interest", which we believe will be useful in an educational setting. For each affect, the representative postures and movements found in the literature are listed along with the analogous behaviors that can be performed by Quori.

At a glance each behavior seems reasonable on its own, but a problem arises when considering the resulting set of Quori behaviors in conjunction with one another. There are several cases of multiple affects sharing similar or identical poses. For example, both happiness and surprise are represented by the torso leaning backward and arms raised high. Similarly, both sadness and interest are represented by the torso leaning forward and arms resting by Quori's side. Thus, these poses currently lead to emotional ambiguity, which is unhelpful for social interaction. We discuss steps for resolving this issue in our section on future work.

IV. EXPERIMENTAL RESULTS

A. Preliminary Gestures

Using our simulated model as a basis, we have implemented a few nonverbal behaviors to showcase Quori's capabilities for social interaction. As seen in Fig. 3, the first of these involves Quori slowly leaning backward while raising its arms above its head, while the second involves Quori quickly leaning forward and stretching its left hand forward. In Table II, the former of these motions may be associated with happiness, while the latter most closely corresponds to anger, but we again stress that the actual perceived emotional state arising from these actions is unknown without user validation. In addition to these two simple behaviors, we have also implemented a more dynamic variant of the first expression, which consists of Quori raising its arms above its head and using its turret joint to quickly spin around 360 degrees. This behavior illustrates two important concepts. First, while there are many motions that can be performed by a human but not Quori, there are some motions that can be easily performed by Quori but not by a human. Secondly, the novelty of dynamic animations and movements performed by a robot could prove effective in provoking interest and fascination from a student, strengthening their willingness to engage with Quori and, consequently, their own learning. However, once again we will require additional work in order to see whether this effect is observed in a real-life setting.

V. FUTURE WORK

In the future, we would like to expand on this work in a few key ways. As mentioned previously, there currently exists some ambiguity in the behaviors we have derived from the literature. Differentiating the expressions of each emotional state from one another as much as possible is important for accurate conveyance of Quori's desired affect. Additionally, we need user validation to determine whether the intended perceived emotional state of each behavior corresponds to its actual perceived emotional state. We intend to conduct a survey asking participants to categorize videos of Quori performing various poses and motions and report the affect they perceive. These videos would include additional variability in the movements by including various combinations of start/end positions, speed, degrees, and asymmetrical motion. The results of this study will not only help us differentiate our behaviors but also validate their effectiveness in conveying emotional states.

Additionally, although we have precise control over Quori's upper-body motions, our control of Quori's wheels and movement is currently limited. We plan to implement a differential drive controller based on the Jacobian to coordinate the base and turret motors to achieve omnidirectional capability. This would allow us to begin developing coordinated motions such as facing a given point in space while moving. Combined with the implementation of cameras, laser scanners, and other sensors for our simulated model, these developments would greatly enhance Quori's capabilities.

Lastly, we would like to implement these behaviors on a real Quori robot and see how they function in a real-world setting. Our Quori has not yet been delivered, so we cannot currently test using real hardware. If effective, these behaviors would be instrumental towards preparing Quori to interact with students in an educational setting. Having been designed with affordability in mind, Quori could one day populate many classrooms to aid in student education.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. I would like to thank Dr. Reid Simmons and Roshni Kaushik for their mentorship, encouragement, and support on this project throughout the summer. I would also like to thank Rachel Burcin and John Dolan for supporting me and my peers over the course of the RISS program.

REFERENCES

[1] S. Thrun, "Toward a framework for human-robot interaction," *Human–Computer Interaction*, vol. 19, no. 1-2, pp. 9–24, 2004.

[2] I. Aaltonen, A. Arvola, P. Heikkilä, and H. Lammi, "Hello pepper, may i tickle you? children's and adults' responses to an entertainment robot at a shopping mall," in *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 53–54.

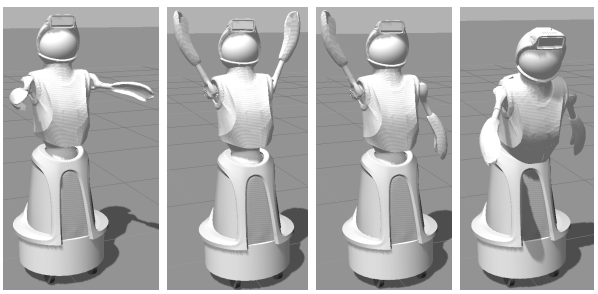


Fig. 3. Non-verbal behaviors performed by Quori in Gazebo

TABLE II
MAPPING MOVEMENTS TO QUORI DOF

Emotion	From Literature	Quori Translation
Happiness	Symmetrical up-down motion of the arms [11] Hands kept high; hands made into fists and kept high [12] Slight lean backwards, arms raised high [26]	Torso leans backward Arms raised high, Arms moving up and down symmetrically
Sadness	Leaning forward, Hands at sides [12], [26] Hands over head [28]	Torso leans forward Arms at sides
Fear	Leaning backward [11] Hands out to sides [12] Body backing, Hands over head, trying to cover body parts [28] (Anxiety) Fingers moving, fingers tapping [28] Leaning backward, Arms slightly forward [26]	Torso leans backward Arms out to sides
Disgust	Leaning backward, Arms forward [26]	Torso leans backward Arms forward
Anger	Leaning forward [11], [12] Arms crossed/on hips [12] Hands on waist, hands into fist or kept low, high speed lift of the hand [28] Leaning forward, Arms forward [26]	Torso leans forward Arms forward
Surprise	Hands over head [28] Leaning backward, Hands over head [26]	Torso leans backward Arms raised high
Interest	Leaning forward, Arms resting at side [11]	Torso leans forward Arms resting at side

[3] D. Albani, A. Youssef, V. Suriani, D. Nardi, and D. D. Bloisi, "A deep learning approach for object recognition with nao soccer robots," in *Robot World Cup*. Springer, 2016, pp. 392–403.

[4] F. Tobe, "How is pepper, softbank's emotional robot, doing?" https://www.therobotreport.com/how-is-pepper-softbanks-emotional-robot-doing/, 2016, accessed: 2020-08-04.

[5] A. Moon, "Nao next gen now available for a wider audience," https://robohub.org/nao-next-gen-now-available-for-the-consumer-market/, 2014, accessed: 2020-08-04.

[6] A. Specian, N. Eckenstein, M. Yim, R. Mead, B. McDorman, S. Kim, and M. Matarić, "Preliminary system and hardware design for quori, a low-cost, modular, socially interactive robot," in *Workshop on social robots in the wild*, 2018.

[7] A. Specian, "About quori," https://www.quori.org/about, 2018, accessed: 2020-07-30.

[8] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[9] W. Garage, "Universal robot description format (urdf)," http://wiki.ros.org/urdf/, 2009, accessed: 2020-07-30.

[10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[11] N. Dael, M. Mortillaro, and K. R. Scherer, "Emotion expression in body action and posture." *Emotion*, vol. 12, no. 5, p. 1085, 2012.

[12] S. Buisine, M. Courgeon, A. Charles, C. Clavel, J.-C. Martin, N. Tan, and O. Grynszpan, "The role of body postures in the recognition of emotions in contextually rich scenarios," *International Journal of Human-Computer Interaction*, vol. 30, no. 1, pp. 52–62, 2014.

[13] F. Tanaka, K. Isshiki, F. Takahashi, M. Uekusa, R. Sei, and K. Hayashi, "Pepper learns together with children: Development of an educational application," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 270–275.

[14] S. Lemaignan, A. Jacq, D. Hood, F. Garcia, A. Paiva, and P. Dillenbourg, "Learning by teaching a robot: The case of handwriting," *IEEE Robotics & Automation Magazine*, vol. 23, no. 2, pp. 56–66, 2016.

[15] N. Lubold, E. Walker, H. Pon-Barry, Y. Flores, and A. Ogan, "Using iterative design to create efficacy-building social experiences with a teachable robot." International Society of the Learning Sciences, Inc.[ISLS]., 2018.

[16] L. Pareto, T. Arvemo, Y. Dahl, M. Haake, and A. Gulz, "A teachable-agent arithmetic game's effects on mathematics understanding, attitude and self-efficacy," in *International Conference on Artificial Intelligence in Education*. Springer, 2011, pp. 247–255.

[17] F. Tanaka and S. Matsuzoe, "Children teach a care-receiving robot to promote their learning: Field experiments in a classroom for vocabulary learning," *Journal of Human-Robot Interaction*, vol. 1, no. 1, pp. 78–95, 2012.

[18] T. P. Mottet and V. P. Richmond, "Student nonverbal communication and its influence on teachers and teaching," *Communication for teachers*, pp. 47–61, 2002.

[19] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017. [Online]. Available: http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf

[20] P. Ekman, "Differential communication of affect by head and body cues." *Journal of personality and social psychology*, vol. 2, no. 5, p. 726, 1965.

[21] P. Ekman and W. V. Friesen, "Head and body cues in the judgment of emotion: A reformulation," *Perceptual and motor skills*, vol. 24, no. 3 PT 1, pp. 711–724, 1967.

[22] A. P. Atkinson, W. H. Dittrich, A. J. Gemmell, and A. W. Young, "Emotion perception from dynamic and static body expressions in point-light and full-light displays," *Perception*, vol. 33, no. 6, pp. 717–746, 2004.

[23] A. P. Atkinson, M. L. Tunstall, and W. H. Dittrich, "Evidence for distinct contributions of form and motion information to the recognition of emotions from body gestures," *Cognition*, vol. 104, no. 1, pp. 59–72, 2007.

[24] J. Montepare, E. Koff, D. Zaitchik, and M. Albert, "The use of body movements and gestures as cues to emotions in younger and older adults," *Journal of Nonverbal Behavior*, vol. 23, no. 2, pp. 133–152, 1999.

[25] F. E. Pollick, H. M. Paterson, A. Bruderlin, and A. J. Sanford, "Perceiving affect from arm movement," *Cognition*, vol. 82, no. 2, pp. B51–B61, 2001.

[26] M. Coulson, "Attributing emotion to static body postures: Recognition accuracy, confusions, and viewpoint dependence," *Journal of nonverbal behavior*, vol. 28, no. 2, pp. 117–139, 2004.

[27] J. L. Tracy and R. W. Robins, "Show your pride: Evidence for a discrete emotion expression," *Psychological Science*, vol. 15, no. 3, pp. 194–197, 2004.

[28] I.-O. Stathopoulou and G. A. Tsihrintzis, "Emotion recognition from body movements and gestures," in *Intelligent Interactive Multimedia Systems and Services*. Springer, 2011, pp. 295–303.

Control for Distributed Dextrous Manipulation on a Multi-manipulator Array

Skye Thompson[1] and Oliver Kroemer[2]

***Abstract*— Distributed manipulators - consisting of a set of actuators or robots working cooperatively to achieve a manipulation task - are robust and flexible tools. Delta arrays, distributed manipulators composed of a parallel grid of 3DOF Delta robots, are proposed as a novel mechanism for performing a range of dexterous manipulation tasks. In this paper, we explore the principles guiding development and control of such a delta array, including optimal arrangement and packing, planar manipulation policies, and cooperative control. Hand-designing effective distributed control policies can be complex and time consuming, given the high-dimensional action space of the manipulator. We examine policy learning as a robust control approach allowing for smooth manipulation of a range of objects, showing improved performance over a baseline human-designed policy.**

***Index Terms*— Distributed Robot Systems, Parallel Robots, Model Learning for Control**

I. INTRODUCTION

Distributed approaches to manipulation, such as those seen in automated conveyors, smart surfaces, and planar manipulators, have long served as valuable tools for use in manufacturing and other domains. Such systems offer an appealing flexibility and resilience in their capabilities, through the redundancy and bandwidth afforded by their many cooperating actuators. But the design and control of distributed manipulators departs significantly from that of more traditional end-effectors. Many are best-suited to non-prehensile and hardware-dependent manipulations with complex, geometry-dependent dynamics - like wheeled conveying, for one example. Designing these kinds of policies is difficult even for manipulating known objects in a controlled environment. Transitioning distributed manipulators to unstructured environments, with unfamiliar objects, requires innovation in both the hardware available and the approach to distributed control.

Individually, a delta robot is a 3 degree of freedom parallel mechanism with a fixed base and moving stage, originally designed for pick-and-place and performing other high-speed, high-precision tasks in factory settings. With a straightforward design and simple kinematics, the delta is an appealing candidate for mass production and coordinated control. Recent advancements have allowed for the production of these robots at smaller scales, making it more feasible to use them cooperatively. By arranging these deltas in a cooperative grid, called a "delta array", it's possible to achieve a wide range of object manipulations, combining a variety of manipulation strategies requiring both stable contact and dynamic cooperation. For example, it's possible to convey an object across the surface of an array with one group of deltas, while raising another group in a wall structure to assist with alignment. Initial experimentation has been promising, but the general principles of design and control for this novel system have not yet been fully explored. We propose to develop these principles in this paper, through examination and experiment at each stage of design and control of the delta array.

We begin by examining the possible configurations of deltas within an array, and the impact each arrangement has on the manipulative capacity of the system. We then explore variations on distributed control approaches to a planar translation task on the delta array. We compare different phase breakdowns of a multiphase finger-gaiting policy and their impact on manipulation performance. We evaluate the impact of individual delta's gait trajectories as well, optimizing for a smooth, efficient translation through policy search.

II. RELATED WORKS

Previous work on the design and control of distributed manipulators has focused primarily on industry applications, and the forms of distributed manipulators most applicable to them. Yaemglin et. al [1] focuses on automated conveyors or other industrial systems. A body of thorough and interesting work on distributed control has been built through these explorations. The approach used by Bedillion et. al relies on dynamics-dependent models of known objects for consistent control [2], while that used by Luntz et. al focuses instead on algorithmic models attempting to guarantee the manipulated object will reach a certain pose, regardless of dynamics [3].

These approaches, while informative, are not clear analogs to those which would be effective on the delta array. Dynamics calculations that may be feasible for a known set of objects interacting with pins, or a surface of spinning wheels, don't translate directly to a system like the delta's with finger-like points of contact with an object, and a different model of interaction requiring closed-form contact or finger gaiting. Additionally, even the most robust of these control approaches, such as the elliptical and squeeze fields explored in Lunz et. al [3] and Bohringer et. al [4], fail to generalize to objects of certain shapes and sizes, for reasons not fully addressed by their system design. From this, we conclude the work of exploring the factors to consider in the design and control of a delta array, or similar distributed manipulator, is vital in guaranteeing it's potential as a flexible, resilient

[1]Skye Thompson is with CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, rsthomp@mit.edu

[2]Oliver Kroemer is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, okroemer@cmu.edu

manipulation system. The unique traits of such a system - the three degrees of freedom of each delta unit, and the difference in models of contact and control between a delta and a wheel or pin actuator - require a novel investigation.

Another body of work is that done on smart surfaces. Barr et. al [5] explores the algorithmic and mechanical properties of a grid of single degree of freedom actuators, capable of rising and lowering to manipulate objects on the surface. The control approach for such a system is still not identical to that of an array of 3DOF robots, like the deltas, which can translate in the XY plane in addition to the Z axis, granting them a wider range of manipulative capabilities, and requiring a different focus of control to execute those skills. Although the dynamics and kinematics of the Delta itself are thoroughly covered in [6], no work is done on the interactions between deltas, or the traits and capabilities of such robots operating in unison.

III. Delta Packing

The first consideration when constructing the delta array is the configuration of the grid of delta manipulators - how tightly to pack them, and in what pattern. This configuration impacts the size and shape of objects that the array can manipulate, as well as what policies work best for manipulating those objects. Our first intuition is that tighter packing, by eliminating gaps between deltas, results in more reliable manipulation of a wider range of objects than a widely-spaced arrangement, as small objects are less likely to fall out of reach of any individual manipulator. However, when packed closely, the legs and stages of neighboring deltas risk colliding, limiting the effective workspace of each, resulting in an inefficient use of available space. An optimal configuration jointly maximizes the density of manipulators in the array, and the effective workspace of each individual delta.

One benefit of the array-like manipulator layout is symmetry, allowing us to extrapolate the workspace of the entire grid from relationships between individual neighboring deltas that are repeated throughout the structure. We evaluated three possible delta packing arrangements for our arrays - square, triangular, and hexagonal, as seen in the figure below. Each of these layouts is composed of only one or two configurations of neighboring deltas that may collide - defined by the rotational and translational offset between deltas. The best packing arrangement is determined as having the neighboring delta configuration with the least potential collisions between deltas.

A. Delta Arrangement

To explore each possible delta arrangement, we developed a set of Matlab simulation tools. The legs, base, and stage of a delta were modeled at a given position and orientation. Each delta was modeled with a 1 cm stage side length, a 1.4 cm base side length, 2 cm long lower legs, and 3cm long upper legs. The maximum possible workspace of each delta was identified by sampling a set of end effector points defining its possible workspace, calculating the inverse kinematic delta configuration needed to reach each one, then eliminating those which violated the angle constraints of the delta's legs. To identify collisions between deltas, two deltas were simulated at a time, at a specified distance and rotational offset from each other. For a given point in each delta's known workspace, the inverse kinematics were calculated to determine the configuration of the legs and stage. A series of collision spheres were then simulated along the length of each leg and the sides of the staging area. Any overlap in these spheres was recorded as a collision. All configurations of the two deltas in their identified maximum possible workspace were tested, and pairs of configurations resulting in a collision were recorded.

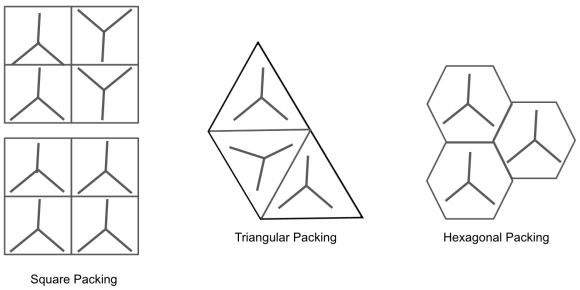


Fig. 1: The possible delta packing configurations associated with each rotational offset. Square packing places neighboring deltas in rows and columns, triangular packing places neighboring deltas at a 60° rotational offset, and hexagonal packing places neighboring deltas in staggered columns

TABLE I: Workspace Median Collisions per Configuration

	Square	Triangular	Hexagonal
Summed Median Collisions	206	100.5	75

The hexagonal packing arrangement clearly demonstrates the least median collisions per point, indicating it was the option that would best preserve the deltas' effective workspace.

As shown in Table I above, results indicated that deltas arranged hexagonally had the fewest collisions, therefore preserving the largest effective workspace. Intuitively, this makes sense - this arrangement places the legs of each delta at an offset from the legs of neighboring deltas, making them less likely to collide. This corresponds with the hexagonal packing method, suggesting that a hexagonal grid of deltas would be the ideal arrangement for efficient use of space.

B. Packing Density and Workspace Volume

Once the best packing arrangement had been identified, our next step was to examine the density, represented by the distance between neighboring Deltas. Each delta's effective independent workspace is defined as the set of points it can reach while guaranteed not to collide with its neighbors. Our goal was to determine how closely neighboring deltas could be placed to each other without sacrificing significant portions of the workspace of each.

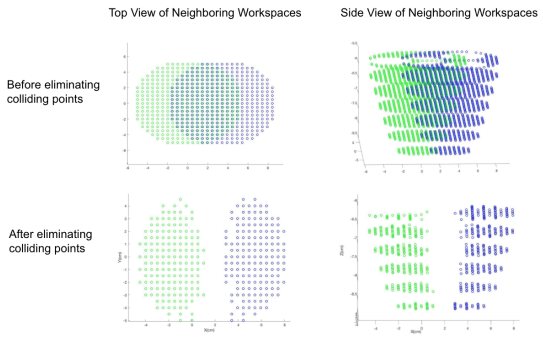


Fig. 2: This figure shows the valid workspaces of neighboring Deltas in a hexagonal arrangement before and after the removal of the fewest possible points to eliminate all collisions.

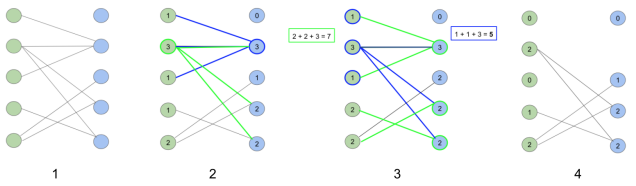


Fig. 3: Collisions were removed using an algorithm which models the points in each delta's workspace as vertices of a graph, and collisions between them as edges. First, the vertices with the most edges - the points with the most collisions - are identified. Next, for each of those vertices, we sum the number of edges of each of their neighbors. We remove the vertex with the fewest secondary edges from the graph, then repeat. This means the point causing the most unique collisions in the graph will be removed.

We examined the pattern of collisions between deltas in a hexagonal packing arrangement, spaced a range of distances apart. When the set of collisions were identified for a given distance and rotational offset, the effective workspace of each delta was determined by eliminating points from the workspace of each delta until no collisions between the two remained. To minimize the points removed to achieve this goal, a simple algorithm was employed, as seen in Figure 3. The points representing a configuration in each delta's workspace were treated as vertices in a bifurcated graph, and the collisions between them as edges. At each step, the vertex with the most collisions, represented as the vertex of highest degree, was removed from the graph. If multiple vertices of the same degree existed, the one with the smallest sum of degrees across its connected vertices would be removed. This incentivized the removal of points uniquely responsible for some set of collisions in the other delta's workspace, preserving the maximum number of points in each. Due to the asymmetry of the hexagonal pattern of deltas, where the leg of one delta would point towards the gap between two legs of its neighbor, collisions were not distributed equally between the two workspaces. It proved important to attempt to conserve a similar number of points in each delta's workspace, to avoid asymmetric, infeasible workspace representations like those shown in Figure 4. This was achieved by imposing an additional constraint on the removal of points; if a workspace had significantly more points remaining than its neighbor (here, a difference of greater than 10 points), only points in that workspace would be considered for removal.

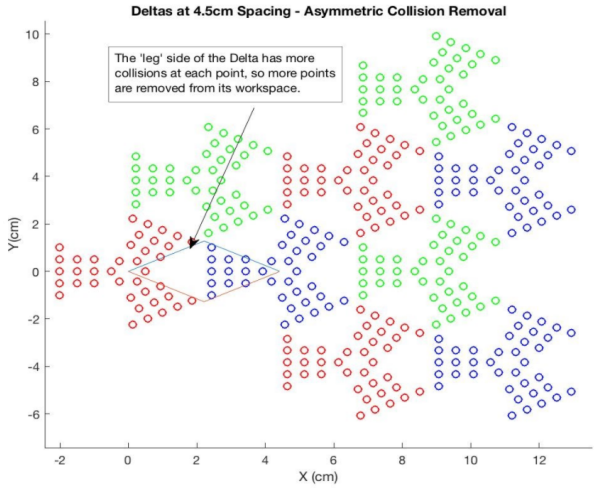


Fig. 4: An example of a delta array workspace constructed without enforcing symmetry between neighboring deltas. The issues with the constructed workspace are due to the different orientation of each delta affecting the number of collisions per point for certain configurations.

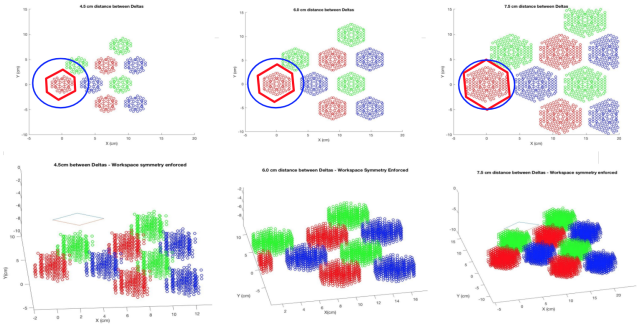


Fig. 5: The size of each delta's workspace approached the maximum possible size as the distance between deltas was increased. Each delta's effective workspace also became more symmetrical as distance increased. Red indicates the size of the delta's effective workspace, while Blue indicates the maximum possible workspace diameter.

The figure above shows how the full accessible workspace of a delta array would evolve as the deltas are spaced further apart. This workspace was constructed by taking advantage of the hexagonal symmetry in the packing arrangement,

tessellating the effective workspace of a single pair of deltas (indicated by the diamond) throughout the grid. The density of each packing arrangement was represented by the surface area of this diamond on the XY plane, while the workspace volume was represented by the number of points falling within that diamond.

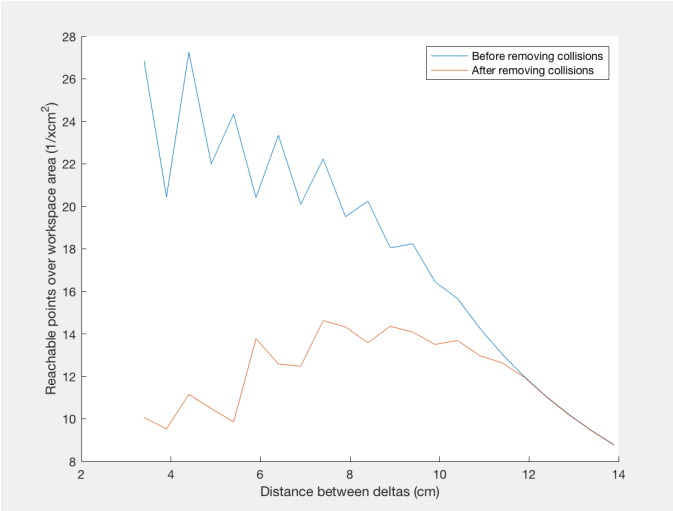


Fig. 6: Here, a joint measure of workspace volume and Delta density are graphed, both before and after removing collisions. After removing collisions, the workspace density (defined as number of non-colliding points divided by the surface area of the packing hexagon) peaked around a distance of 7.5 cm, where the effective workspace of each delta reaches the maximum possible workspace with no significant gaps between deltas.

From these evaluations, we can conclude that a hexagonal packing of deltas is the most feasible and effective arrangement for a manipulator array, and that the density of that packing does significantly impact the effective workspace of each individual manipulator. Packing arrangements up to 7.5 cm distance between neighboring deltas, for the modeled specifications, can provide sufficiently complete coverage of the array workspace, while greater distances result in growing gaps between deltas that create unreachable space. Even up to that distance, however, different densities may result in different capacity to manipulate objects of a range of sizes. In order to continue this evaluation, we move to an examination of basic planar manipulation techniques.

IV. Translation Analysis

Planar manipulation is the primary realm of many common forms of distributed manipulation, like automatic conveyors or smart surfaces - simply translating or rotating objects in the XY plane. Due to the 3 degrees of freedom of each individual manipulator, a delta array has an additional ability to perform three-dimensional manipulation skills, like grasping. But planar manipulations can both contribute to useful three-dimensional skills, like re-grasping, as well as provide insight on how the delta array's design can impact its general manipulative capacity. We want to be able to translate and rotate objects in plane. Ideally, these manipulations will also be smooth, quick, and consistent. However, because such manipulations are non-prehensile, object behavior can be heavily dynamics-dependent.

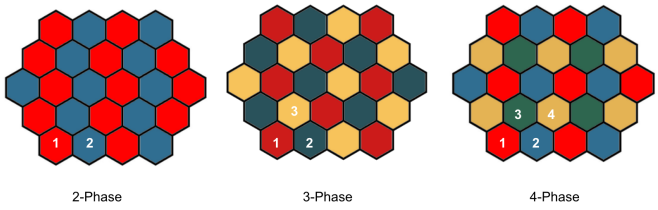


Fig. 7: The three possible phase breakdowns of a translational or rotational gait on a hexagonal grid.

One straightforward approach to planar manipulation is the use of multiphase finger gaiting, where separate groups of deltas move together in a staggered pattern, transferring the object between the deltas assigned to a given phase as they oscillate. For a hexagonal packing, there are three potential phase breakdowns for a given manipulation - dividing the deltas into two, three, or four groups as seen below, where deltas marked by the same color move together. A translation using more phases may lead to a smoother trajectory in the desired direction, as the handoff between phases occurs before the current face stops or begins to reverse. But with more phases, fewer deltas are in contact with the object at any given time, which may contribute to instability in the object trajectory, particularly for smaller objects. Our next goal is to determine how the use of different phases impacts the path and behavior of a translated object, and to establish what constraints the use of each phase breakdown imposes on the objects that can be successfully manipulated.

A. Phased Gait Evaluation

We constructed a simulated, 8x8 delta array in Coppeliasim. Each delta manipulator is represented by a hexagonal base and a cylindrical end-effector, controlled by prismatic x-, y-, and z-joints to replicate the delta's 3 degrees of freedom. A flat, cuboid object served as the manipuland, placed on top of the delta array. For this analysis, we assume each delta moves in a simple elliptical pattern with constant speed, in the desired direction of translation, with a phase delay corresponding to the phase breakdown of the maneuver, and the position of the delta on the grid. The path of each delta can be described as:

$$x_t = d_1 \cdot cos(\theta) \cdot cos(\phi + t \cdot \frac{\pi}{f})$$

$$y_t = d_1 \cdot sin(\theta) \cdot cos(\phi + t \cdot \frac{\pi}{f})$$

$$z_t = d_2 \cdot sin(\phi + t \cdot \frac{\pi}{f})$$

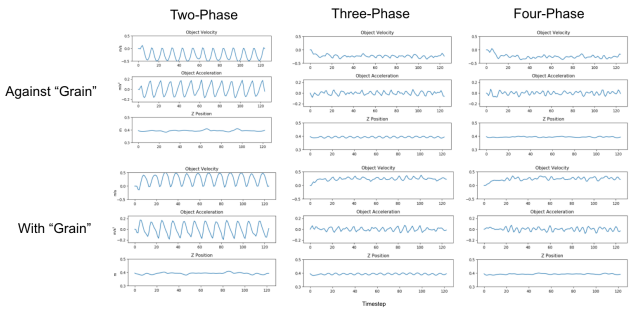


Fig. 8: Details on the object trajectory for a planar translation skill using each possible phase breakdown. Higher-phase gaits showed smoother travel, with lower average acceleration and Z-axis deviation of the translated object, regardless of the direction of the translation.

Where d_1 and d_2 are the major and minor diameters of the ellipse, θ is the angle of translation, and ϕ and f are the phase delay and frequency of the oscillation respectively.

For our first evaluation, we compared 2-, 3-, and 4-phase gaits for translating a square object of fixed side length across the surface of the array. We compared between translations parallel and perpendicular to the "grain" of the phase breakdown in each, as shown in 8. We found that gaits with more phases tended to produce smoother trajectories, with less vertical deviation, and a lower average acceleration, likely due to the smaller phase delay between each phase resulting in a more constant velocity. However, the increased spacing between Deltas in the same phase caused a different form of instability in the trajectory. With the higher-phase gaits, the object was more likely to tip due to having fewer points of contact with the array.

To explore this further, the translation test was repeated with objects of varying side length. The vertical deviation, acceleration, and maximum roll, pitch, and yaw deviations of the object across the trajectory were measured. Performance on all gaits stabilized when the surface area of the object to be translated was approximately $1.44m$, about 8 times the area of the delta's simulated hexagonal workspace ($.175m$). This suggests a lower bound on the surface area of manipulable objects, and indicates that although increased distance between deltas may be more space-efficient, as explored in the first section, the range of objects such an array can manipulate may be reduced at higher distances.

V. Gait Trajectory

Having examined how the design of a delta array may impact it's manipulative capacity, we now delve into its control. We previously assumed that each individual delta would perform a simple oscillating elliptical gait - but the optimal gait may differ significantly from this prior, and may be different given different skills. To avoid the difficulty of hand-evaluating the possible dynamics of each interaction, we can approach this problem with learning. One example to explore is a comparison of the effective trajectory for different phased translational gaits. To improve the efficiency of manipulation using the delta array, we aim to learn an improved trajectory that will translate an object quickly and smoothly.

We aim to learn this improved trajectory through episodic Relative Entropy Policy Search, or REPS [7], a policy search approach that imposes a constraint on the similarity of each updated policy distribution to its predecessor. Our lower-level control policy $u(t)$ is represented as a series of weighted Gaussians, centered at N equally spaced timesteps. At any timestep, the target position of the Delta is calculated as $s_t = \frac{\sum_i^N \omega_i \mathcal{N}(\mu_i,\sigma_i)}{\sum_i^N \mathcal{N}(\mu_i,\sigma_i)}$. This allows us to parameterize a smooth trajectory with a smaller number of points, N, for efficient learning.

For a given episode, our higher-level policy, $\pi_\omega(\theta)$, consists of an N-dimensional Gaussian representing the weights used in the lower-level policy $u(t)$. A set of weights, $w_{1..N}$, are sampled from this Gaussian at the beginning of the episode, and used to define the cyclical trajectory to be executed. Each episode consists of a rollout of this policy across a fixed number of timesteps. The reward signal, $R(\theta) = \frac{1}{z_1}v - \frac{1}{z_2}h - \frac{1}{z_3}a$, obtained at the end of this rollout, considers the average speed of the object in the desired direction v, its maximum vertical deviation h, and its maximum acceleration a, where z_{1-3} are normalizing factors.

The higher level policy is updated every epoch, and the new distribution $pi_\omega(\theta)$ is approximated by performing a maximum-likelihood estimate on the samples $\theta^{[i]}$, weighted by weights $d_i = exp(\frac{R(\theta^{[i]}}{\eta})$. The parameter η is found by minimizing the dual function:

$$g(\eta) = \eta\epsilon + \eta log(\sum_i \frac{1}{N}\left(\frac{R(\theta^{[i]})}{\eta}\right))$$

where ϵ is the upper bound established on the KL-divergence between the current distribution and the new distribution.

We found the learned policy for all phases, after 30-50 epochs of 5 episodes each, converges to a trajectory with a widened, flattened upper half compared to the elliptical prior. This flat period is shorter for higher-phase gaits, where the delta spends less time in contact with the object.

VI. Conclusion and Future Work

In our experiments, we identified several principles and tradeoffs to guide development of the delta array manipulator, including optimal arrangement and packing, planar manipulation policies, and learned control. We present these to inform further use and study of such systems. One next step would be to explore further how learning can inform policy design for this type of manipulator, and distributed manipulators more generally. Efficient and effective high and low level control policies can be difficult to identify in complex systems, but the preliminary approaches to learning examined here suggest that it may be possible to learn such policies, and achieve better performance with the manipulator. Another important future pursuit is a more detailed

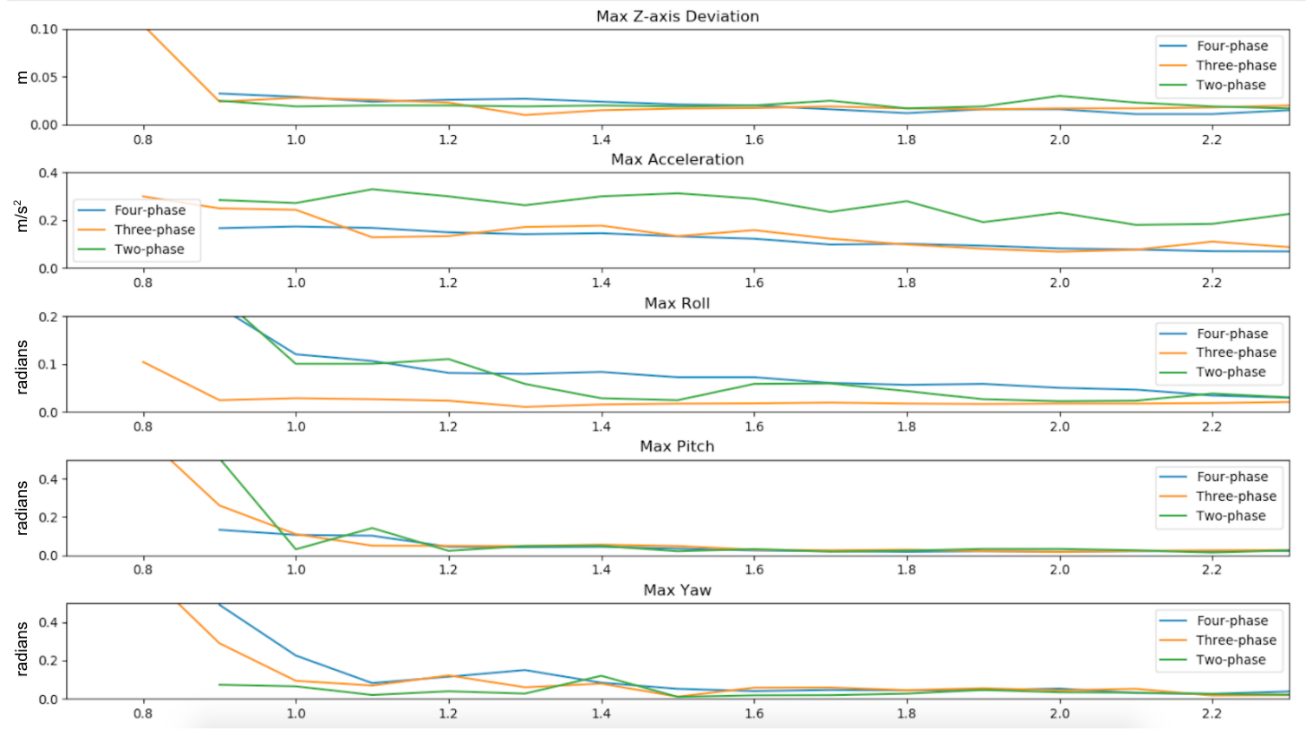


Fig. 9: A comparison of the performance of each phased gait on objects of different side length. Performance on all gaits stabilized when the surface area of the object to be translated was approximately 1.44m, about 8 times the area of the delta's hexagonal workspace in simulation(.175m).

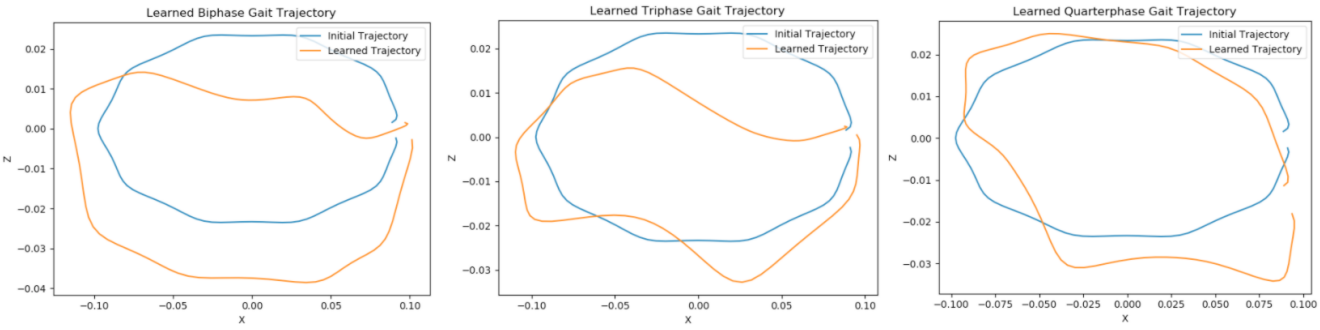


Fig. 10: The learned gait trajectory for each of the possible phase breakdowns. While the bottom half of the trajectory, where the delta is no longer in contact with the object, is highly variable, the period of contact is consistently smooth and slightly sloped across all the trajectories.

consideration of the dynamics of the delta robots in the array. The force and compliance each delta is capable of generating could have a significant impact on the array's manipulative capacity.

ACKNOWLEDGMENT

Special thanks to the RISS program, the Robotics Institute at CMU, and the Intelligent Autonomous Machines Lab for their support. This material is based upon work supported by the National Science Foundation under Grant No. 1659774.

REFERENCES

[1] T. Yaemglin and S. Charoenseang, "Distributive behavior-based control for a flexible conveying system," *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT '02*, pp. 24–29 vol.1, 2002.

[2] M. Bedillion, R. Hoover, and J. McGough, "A distributed manipulation concept using selective braking," *2014 American Control Conference*, pp. 3322–3328, 2014.

[3] J. E. Luntz, W. C. Messner, and H. Choset, "Distributed manipulation using discrete actuator arrays." *The International Journal of Robotics Research*, pp. 20:553–583, 2001.

[4] K. Bohringer, B. Donald, R. Mihailovich, and N. C. MacDonald, "A theory of manipulation and control for microfabricated actuator arrays," *Proceedings IEEE Micro Electro Mechanical Systems An*

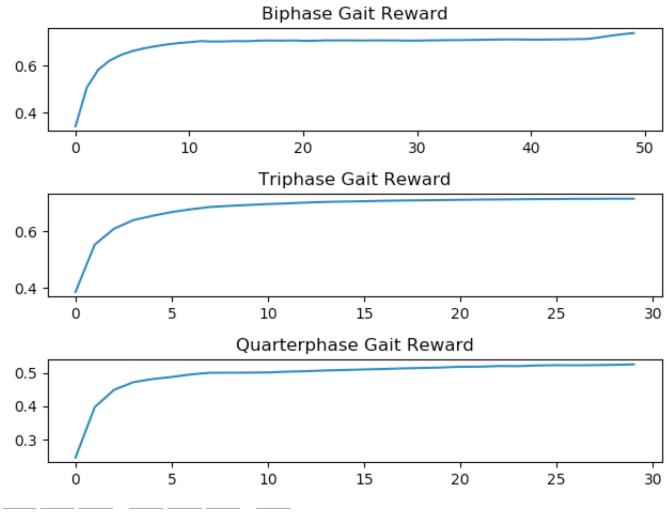


Fig. 11: A comparison of the learning convergence over the three phase breakdowns.

Investigation of Micro Structures, Sensors, Actuators, Machines and Robotic Systems, pp. 102–107, 1994.

[5] D. R. W. Barr, D. Walsh, and P. Dudek, "A smart surface simulation environment," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4456–4461, 2013.

[6] M. Lopez, E. Castillo, G. Garcia, and A. Bashir, "Delta robot: inverse, direct, and intermediate jacobians," *Journal of Mech. Eng. Science*, pp. 220:103–109, 2006.

[7] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, pp. 57–58:130–136, 2013.

[8] M. Bedillion and W. Messner, "Trajectory tracking control for actuator arrays," *IEEE Transactions on Control Systems Technology*, pp. 21(6):2341–2349, 2013.

[9] K. F. Bohringer, H. Choset, and H. Choset, *Distributed Manipulation*. Springer Science Business Media, 2000.

[10] M. Sinclair and I. A. Raptis, "Distributed manipulation using cyber-physical systems," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3097–3102, 2014.

Effective Collision Avoidance System for Unmanned Aerial Vehicles

Fausto Vega[1], Mohammadreza Mousaei[2], and Sebastian Scherer[2]

***Abstract*— Collision avoidance for unmanned aerial vehicles is critical to ensure safety in autonomous missions. Several methods have been implemented to ensure a safe set around an aircraft, yet they have not been set as a global standard such as the Traffic Collision Avoidance System (TCAS) system for large aircrafts. This paper will present a new collision avoidance method using boundary certificates and compare its performance to the state-of-the-art methods that are currently being explored. Airborne Collision Avoidance System (ACAS) sXu was simulated through the XPlane-11 flight simulator to visualize the detect and avoid algorithm as well as analyze the advisories given by the algorithm. A control barrier function approach was also investigated and validated on a drone platform which provided promising results as the drone only deviated when required.**

***Index Terms*— Aerial Systems: Applications, Collision Avoidance**

I. INTRODUCTION

Unmanned aerial vehicles (UAV's) have begun populating the airspace at high rates due to their versatile form factor that allow robust performance in various applications. From disaster response, to agriculture monitoring, UAV's increase efficiency by automating processes and capturing data via onboard sensors for further processing. With an increasing amount of air traffic, an effective collision avoidance system needs to be established to ensure safety among all vehicles. Currently, large commercial vehicles utilize a Traffic Collision Avoidance System (TCAS) to safely navigate the airspace by a series of radio signals via onboard transponders. From these signals, instructions are given to both aircrafts if a possible collision threat is detected [1]. In terms of UAV's, obstacle avoidance sensors exist such as vision and LiDAR, yet a framework to connect multiple vehicles to one collision avoidance system continues to be explored. State of the art collision avoidance systems for UAV's include a probabilistic model that use Markov Decision Processes to select the best policy based on the encounter. Yet these methods are still in progress and have not become a global standard like TCAS. UAV's also contain certain restrictions such as having it fly within a line of sight of the user as well as strict control of drones in urban areas [2]. With an integrated collision avoidance system, these vehicles can be managed efficiently and perform automated missions in any indoor/outdoor space. TCAS has been experimented with a UAV platform however it is unreliable as the antenna utilized in TCAS does not provide an accurate bearing [3]. TCAS is designed for commercial aircrafts with the assumption of an onboard pilot that can interpret the display; therefore,

Fig. 1. XPlane-11 flight simulator window

UAV's need a new system to effectively navigate the air space [4]. Several approaches have been developed to address this problem such as a geometric approach that considers UAV's as point mass and calculates the point of closest approach to evaluate the worst conflict among UAV's [5]. Another method developed by Lin includes a sampling-based path planner for UAV collision avoidance and validated it with indoor and outdoor tests [6]. A velocity obstacle method was also used to calculate the optimal safe velocity of an agent, yet this method was not suitable for highly dynamic collision avoidance [7]. The performance of these methods has not been compared to state-of-the-art collision avoidance systems to validate performance results. With these methods onboard, it expands the capabilities of a UAV as it can lead to multi-agent collaboration to cover a large area as well as autonomous missions for applications such as a drone delivery system.

In this paper, a collision avoidance scheme for UAV's will be presented and will be compared to the current state of the art collision avoidance systems in aviation. This system will utilize barrier certificates which will ensure collision free behaviors in multi-robot systems. The method of barrier certificates utilizes control barrier functions which defines several safety barrier constraints that will be minimized [8]. An ideal flight controller must be minimally invasive which means it should only notify the vehicle when a collision is evident. TCAS sends notifications when a plane is within a certain range yet does not take aircraft heading into account which leads to unnecessary notifications. The current state of the art collision avoidance system will be simulated using the X-Plane 11 flight simulator to provide an accurate test environment. Metrics such as warnings alerts and distance

[1]Fausto Vega is with Department of Mechanical Engineering, University of Nevada, Las Vegas, 4505 S. Maryland Pkwd, Las Vegas, NV 89154 vegaf1@unlv.nevada.edu

[2]Mohammadreza Mousaei and Sebastian Scherer are with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA {mmousaei,basti}@cmu.edu

between aircrafts will be observed to confirm the most efficient system.

The remainder of the paper is organized as follows: Section II describes the methods taken to simulate the ACAS sXu algorithm on XPlane-11 as well as the drone specifications and software to simulate the control barrier function approach. Section III will present the results of both experiments, and Section IV will address conclusions and future work on this project.

II. Methods

A. XPlane-11

XPlane-11 is a flight simulator that displays accurate physics simulations of various aircrafts and provides detailed scenery with the option of multiple weather conditions. This platform allows the user to select from a variety of aircrafts, each with unique dynamics, sensors, and controls. The performance and handling of the aircraft are predicted, and XPlane-11 allows the user to test control algorithms and trajectories for the aircraft. The aircraft chosen for this project was the Taranis unmanned aerial vehicle that is developed by BAE Systems as it fits the dimensions of the collision avoidance system that was investigated. The aircraft interior in the XPlane-11 environment is shown in Figure 1.

XPlane-11 publishes several values called data refs via UDP communication. These values specify the current state of the aircraft in terms of controls, dynamics, and onboard instrumentation. XPlane-11 provides a software development kit (SDK) which allows users to develop additional programs (plug ins) to modify the XPlane-11 environment and to collect data. AI aircrafts (aircrafts controlled through the XPlane-11 software) are also available to provide air traffic, test ownship instrumentation, and collision avoidance systems which was the main focus of the project.

1) XPlane Connect: The X-Plane Connect Toolbox was the research tool used to communicate with the simulator as it allowed the user to receive state information and program the control surfaces of the aircraft. This tool contains several clients and languages to interface with the simulator by reading and writing to any data ref available on XPlane-11. The specific datarefs used in this project are as follows:

- sim/flightmodel/position/longitude
- sim/flightmodel/position/latitude
- sim/flightmodel/position/elevation
- sim/flightmodel/position/local_vx
- sim/flightmodel/position/local_vz
- sim/multiplayer/position/plane1_lat
- sim/multiplayer/position/plane1_lon
- sim/multiplayer/position/plane1_v_z
- sim/multiplayer/position/plane1_v_x

B. Framework

1) ACAS sXu: ACAS sXu is the collision avoidance system being tested which is a computational method that produces optimized decision logic for the aircraft. This method is current research in airborne collision avoidance for manned and unmanned vehicles at the MIT Lincoln Laboratory. The output of this logic is a surveillance and tracking module (STM) that detects an aircraft and tracks position, and a threat resolution module (TRM) that identifies aircraft threats and resolution guidance. The algorithm input requirement is the state of both the ownship and the intruder aircraft to generate the advisories and the resolutions. The state of both aircrafts was done using the datarefs listed above.

Figure 2 shows the flow chart of the data transfer methods from XPlane-11 to ACAS sXu. The plug in, X-Plane Connect, gathers the data from XPlane which is then transferred to a ROS publisher that is publishing at a rate of 50 Hz. ACAS sXu was modified to be ROS compatible and subscribe to the data to generate the reports. As the threat resolution report was generated as a JSON file, it was parsed to read the resolution outputs. A set of codes are provided by ACAS that signify horizontal and vertical advisories from the JSON file. From this information, the control surfaces of the aircraft are commanded to avoid the collision path.

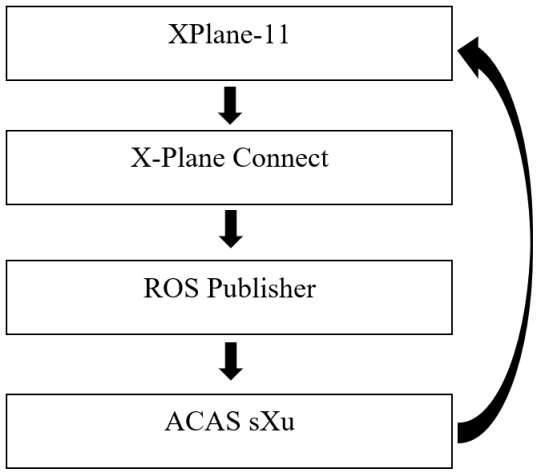


Fig. 2. XPlane-11 Simulation Flow Chart

C. Simulation Experiment

The ownship aircraft was programmed to take off from an airport and once it reached a safe height, the autopilot datarefs were engage to fix the pitch and velocity of the aircraft. Once the ownship was stable in the air, an AI aircraft was spawned at 10 m east and 50 m north of the aircraft to generate a collision path and engage the collision avoidance algorithm. As the aircraft spawned, the threat resolution module report generated a descend advisory which is sent to the aircraft, and the control surfaces are commanded to act accordingly. Figure 3 shows an AI aircraft spawned in the scene which generates the collision advisory from the ACAS system. The red circle is the ownship aircraft while the black circle in Figure 3 is the AI aircraft that is spawned.

Fig. 3. AI aircraft spawned in XPlane scene

D. Control Barrier Functions

Control barrier functions are another form of collision avoidance that impose safety constraints on the trajectory of an aircraft to ensure safety. A safe set is pre defined as the aircraft traverses through the air space and the safety constraints control the safe set of the aircraft. A function h is defined and it maps n dimensional state space to one value which compresses the notion of safety. Equation 1 expresses a control barrier function which is an inequality that ensures the safety of the aircraft as it takes actions. In Equation 1 f and g are the dynamics of the quadroter in control affine form. As long as this notion of safety h is greater than a certain constant, then the aircraft is guaranteed to be in the safe set.

$$\dot{h}(x,u) = \frac{dh}{dx}(x)(f(x) + g(x)u) \geq -\gamma(h(x)) \qquad (1)$$

1) Hardware Integration: A drone platform was developed to validate the control barrier function algorithm simulation. A 70 cm plastic drone frame was connected with four brushless motors and electronic speed controllers to generate the overall thrust of the system. The Pixhawk 4 was the autopilot system used to control the drone autonomously, yet it was configured to work in a GPS denied environment for use indoors. For localization, an Intel T265 sensor was used because of its small form factor and simultaneous localization and mapping capabilities on board the sensor. A Rapsberry Pi 3 was the main source of hardware as it combined the sensor and the flight controller under a unified system.

2) Software Integration: The Raspberry Pi 3 was configured to run Ubuntu 18.04 in order to utilize ROS melodic to control the drone. The librealsense package was installed on the Raspberry Pi 3 to provide the necessary drivers for the tracking camera, and the realsense-ros package was also downloaded for the sensor to publish data streams through ROS topics. For the pixhawk to communicate with the Raspberry Pi 3, a serial to USB cable was made to transfer the data, and the mavros package was installed to ensure

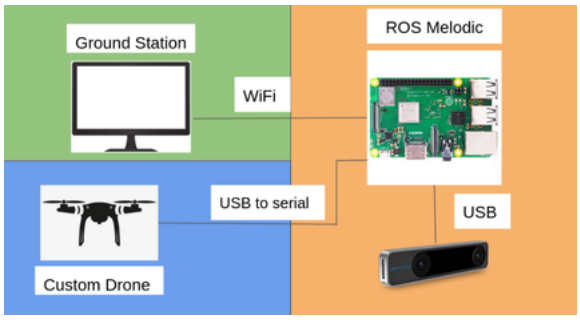


Fig. 4. Hardware Integration of Drone Platform

communication through the MAVLink communication protocol. From the simulation, a ROS node published the position of the aircraft in the frame to the setpoint_position/local topic from mavros. Once mavros received the topic, the drone was switched to offboard mode and followed the waypoints generated through the simulation. A comparison of these images are shown in section III.

3) Experiment: A drone was placed at the (0,0) position in a cartesian coordinate system centered in the middle of the workspace. An obstacle was placed 0.5m to the right and 1.5 m in front of the drone. The goal position of the drone was tasked at 1.5 m to the right and 2.5 meters forward. From the simulation, the waypoints of the drone were divided by 2 due to the limited workspace. Figure 5 shows a diagram of the experiment setup.

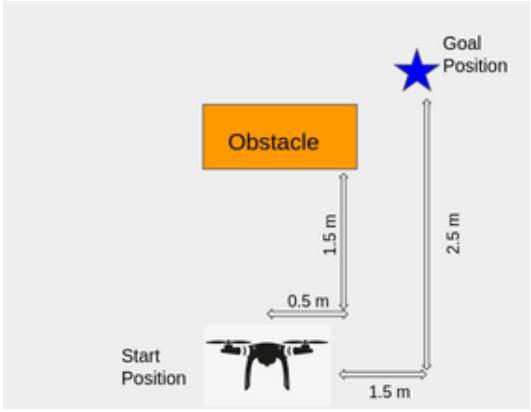


Fig. 5. Experiment setup with dimensions

III. Results

The drone followed the waypoint path generated from the simulation and only deviated from the obstacle when necessary. A safety distance of 0.75 meters was set to prevent collisions. The waypoint data from the simulation to the drone was transmitted at 40 Hz to allow the drone to move effectively to each point. A common fault that arises in the control barrier function algorithm is the issue of deadlock as the robot stays in a constant position trying to determine the safe set. Figure 6 shows the control barrier function simulation plotted using matplotlib. The red ellipse represents the robot, the green ellipse is a safety distance, the

point is the goal position, and the grey is the obstacle set in the simulation. To match the conditions of the simulation, the setup from Figure 5 was established and the drone platform in Figure 7 shows the test in progress at the same timestep as the simulation.

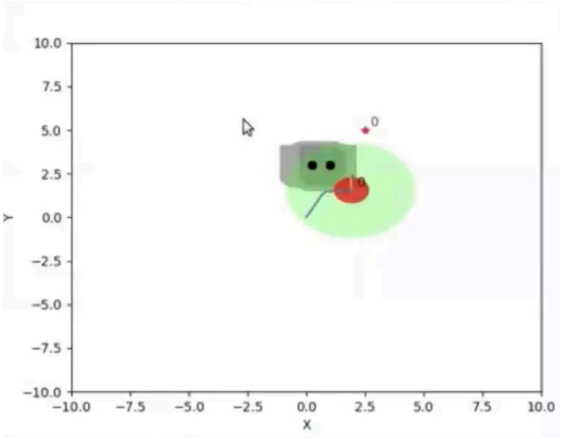


Fig. 6. Control Barrier function simulation

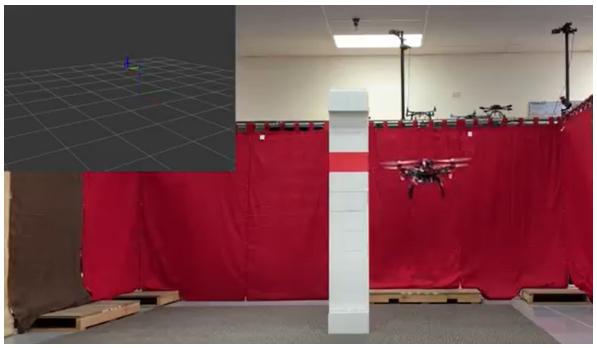


Fig. 7. Drone platform experimentation

IV. Conclusion

This work presented a simulation of a state of the art collision avoidance system called ACAS sXu on the XPlane-11 flight simulator by transferring aircraft data via ROS. A control barrier function approach towards collision avoidance was also presented and validated on a drone platform to ensure the reliability of the algorithm on real world systems. Collision avoidance for unmanned aerial vehicles presents an essential task to achieve safe air traffic control among all systems. This is also crucial for indoor applications of aerial vehicles such as inventory management in which a drone has to navigate across a warehouse and avoid obstacles in its trajectory.

Future work includes testing a control barrier approach with aircraft dynamics instead of utilizing the control affine dynamics used in this work. XPlane-11 can provide these accurate dynamics and a comparison to ACAS sXu in terms of specified metrics such as distance between vehicles and number advisories can be recorded. Currently, the control barrier approach only generates one solution, therefore future work also includes generating a greater amount of solutions and forward simulating several time steps to create a set branches that will be optimized to select the best action path. The final goal includes testing the modified control barrier function approach on small drone platforms such as the Mavic Air.

V. Acknowledgments

I would like to thank the Carnegie Mellon Robotics Institute for the opportunity to work in the Air Lab as well as my mentors Mohammadreza Mousaei and Dr. Sebastian Scherer for providing guidance and perspective through out the project. Also, I appreciate the efforts of Ms. Rachel Burcin and Dr. John Dolan for organizing and allowing the RISS research experience to continue virtually. I would also like to thank Mobility 21 for sponsoring the work done this summer.

References

[1] W. Harman, "TCAS: A system for preventing midair collisions."
[2] UTM Research Transition Team Sense and Avoid Working Group, "Technical work package #2: UTM conflict management model." NASA, FAA, 2018.
[3] J.-W. Park *et al.*, "UAV collision avoidance based on geometric approach," in *SICE Annual Conference*. Korea Advanced Institute of Science and Technology, 2008.
[4] J. K. Kuchar, "Safety Analysis Methodology For Unmanned Aerial Vehicle (UAV) Collision Avoidance System."
[5] H.-C. Lee, "Implementation of collision avoidance system using TCAS II to UAVs," in *IEEE Aerospace and Electronic Systems Magazine*. Korea Aerospace Research Institute, 2006.
[6] Y. Lin, "Sampling-based path planning for UAV collision avoidance," in *IEEE Transactions on Intelligent Transportation Systems*, 2017.
[7] J. van den Berg *et al.*, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3475–3482.
[8] L. Wang *et al.*, "Safety barrier certificates for collision-free multi-robot systems," in *IEEE Transactions in Robotics*. Georgia Institute of Technology, 2017.

Planning with Online Topological Memory

Eliot Xing[1], Sam Powers[*2], Victoria Dean[*2], and Abhinav Gupta[2]

***Abstract*— Reasoning over long horizons is a difficult problem for reinforcement learning that is usually addressed by introducing hand-engineered reward signals or brittle temporal hierarchies. Recent work in long-horizon navigation has combined planning with reinforcement learning by constructing a graph offline on sampled environment observations and using graph search to plan subgoals. This decomposes a long-horizon goal into a sequence of shorter horizon subgoals. However, these methods do not use planning while training the policy, making it difficult to apply them to manipulation tasks where policies execute more complex actions. We extend this work by proposing online topological memory, which dynamically constructs a graph on prior experience during training. Our method uses planning during training to set intrinsic subgoals for the agent that guide exploration. We use reinforcement learning to train a goal-conditioned local policy that can execute a plan of short-horizon subgoals, while offloading long-horizon, global reasoning to graph planning.**

***Index Terms*— Reinforcement Learning**

I. Introduction

We want to build agents that can perform long-horizon, goal-based tasks, such as searching through cabinet drawers in a kitchen to find a certain cup. Deep learning based approaches enable us to handle high-dimensional inputs such as images, and deep reinforcement learning has found success in learning control policies for robotic tasks. However, reinforcement learning algorithms fail to reason over long horizons, even when the inputs are ground truth, low-dimensional state observations. In the absence of dense, human-specified reward functions, model-free reinforcement learning algorithms fail to effectively explore the state space, and can converge to suboptimal fixed points [1]. Rather than trying to directly achieve goals for a given task, an agent should set its own subgoals and learn in a more sample-efficient manner.

Prior work has introduced methods for offline construction of a graph on observations in simulated visual navigation tasks [2]–[4]. However, without constructing a graph online, these methods cannot plan subgoals while training the policy. This makes it difficult to apply these methods to tasks like manipulation which require learning more complex policies that consider the environment dynamics, compared to navigation tasks.

To this end, we propose online topological memory, which builds a graph on observations online, and plans on the graph to generate sequences of subgoals that guide exploration. We show that online topological memory can be used with planning to solve goal-reaching tasks with improved sample-efficiency.

*Equal contribution

[1]Eliot Xing is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA., USA. exing@gatech.edu

[2]Sam Powers, Victoria Dean, and Abhinav Gupta are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA., USA. {snpowers, vdean, gabhinav}@andrew.cmu.edu

II. Related Work

A. Goal-conditioned reinforcement learning

Goal-conditioned policies [5]–[7] take actions to reach a specified task goal given the current state. Goal-relabeling [8] has been shown to improve the sample-efficiency of training goal-conditioned policies [9]–[13]. Our method trains a goal-conditioned policy and applies goal-relabeling by checking nodes in the graph.

Several works have explored learning distance functions between goals [3], [14], [15] to guide learning. Goal-conditioned policies can also be combined with planners [16] to solve longer-horizon tasks. We use graph search with Dijkstra's as the planner to generate a path of subgoals for the goal-reaching policy.

Following [17]–[19], we use a goal-conditioned policy to return to a state that has already been visited and should be reachable by the policy, and then begin stochastic exploration. This can viewed as inducing a curriculum [20], [21], as the local policy must have learned to consistently reach the planned subgoals before attempting the desired, long-term task goal. The planner selects these subgoals by treating the graphical memory as a non-parametric generative model. Maximum-entropy methods [12], [13] have also been proposed to set intrinsic goals for exploration.

B. Topological memory

A number of recent works in simulated visual navigation have used graph-based planning with goal-reaching policies by constructing a graph on observations. Savinov et al. [2] build a graph given a human-provided walkthrough demonstration, with edges determined by a Siamese network that predicts whether nodes are temporally close. Eysenbach et al. [3] learn distributional Q-values as distances for edge weights and assume that the environment can be uniformly sampled. Laskin et al. [4] build a sparse graph on trajectories obtained by random exploration, subsampling edges with k-nearest filtering and proposing a two-way consistency check to merge nodes using an asymmetric distance function. These prior methods construct a graph offline for inference, while our algorithm dynamically builds a graph during training to guide learning. Chaplot et al. [22] use supervised learning to incorporate geometric and semantic properties and update a graph during testing to navigate in unseen house environments. Deng et al. [23] present dynamic graph

construction using graph neural networks for visual and language navigation.

Several studies have also evaluated using topological memory beyond visual navigation tasks, with model-free [24]–[26] or hierarchical [27]–[30] reinforcement learning. These works propose different methods to divide the goal space [25], [27]–[30] and reduce the number of nodes stored in the graph, to guide the agent to unexplored states using the graph [25], [27], [28], or to propogate Q-values between states connected in the graph [24], [26].

III. Preliminaries

We consider goal-reaching tasks for an agent interacting in an environment. Each episode starts by sampling a start state and goal pair from some distribution $\rho(s_1, g)$. For simplicity, let the goal space $\mathcal{G}$ be the entire state space. At each timestep t, the agent observes its current state $s \in \mathcal{G}$ and goal $g \in \mathcal{G}$. The agent then samples an action $a \sim \pi(a|s, g)$ and receives a reward r_g equal to 0 if the goal is reached and -1 otherwise.

The objective of learning is to maximize the expected return $R_t = \mathbb{E}[\sum_{i=t}^{\infty} \gamma^{i-t} r_i]$ under the environment dynamics and current policy, with discount factor $\gamma \leq 1$. A goal-conditioned Q-function learns $Q^\pi(s_t, a_t, g) = \mathbb{E}[R_t|s_t, a_t, g]$. Under the sparse reward r_g with $\gamma = 1$ and a finite-horizon, the absolute value of the Q-function gives the expected number of steps for the policy to transition from state s_t to goal g.

We train a goal-conditioned policy using off-policy model-free reinforcement learning algorithms: DDPG [31] for continuous action spaces and DQN [32] for discrete action spaces. We learn a distributional value function [33] following [3] for the given sparse reward r_g. Unlike [3], for a given maximum Q-value N, we learn a set of $N+1$ bins, $B = (B_0, B_1 \ldots, B_N)$, where B_i corresponds to a given state and goal being i steps away.

IV. Online Topological Memory

In this section, we introduce our method of online topological memory. We begin by describing the neighborhood function and online graph construction. Then, we discuss two ways the agent can use the graph during training, by planning for exploration and goal-relabeling using the graph.

A. Neighborhood-based partitioning

We build our topological memory using a neighborhood function b which partitions the goal space into regions. We consider two different sets of thresholds, b_{merge} and b_{reached}. We use b_{merge} to determine if nodes should be merged together during online graph construction and b_{reached} to check if a subgoal has been reached during a trajectory.

For goal spaces that are Euclidean domains, we assume that the l^2-norm gives an appropriate distance measurement between states. For example, in navigation tasks, this corresponds to coordinate positions in the environment. In practice, for these state-based environments, we use $b_{\text{merge}} = b_{\text{reached}}$.

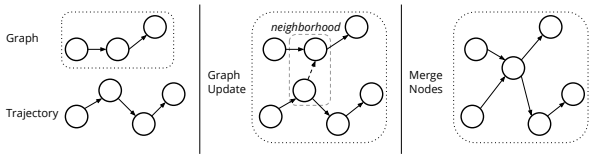


Fig. 1: Diagram of online graph construction using our method. If states in a trajectory are localized as nodes in the graph, then the nodes are merged. Otherwise states are added as nodes.

B. Online graph construction

Algorithm 1 Online Topological Memory

Require: embedding ϕ, neighborhood function b, replay buffer $\mathcal{R}$, local policy π, value function D

vertices $V \leftarrow \emptyset$, edges $E \leftarrow \emptyset$, graph $G \leftarrow (V, E)$
for $n = 1 \cdots N$ episodes **do**
  $T \leftarrow$ collect trajectory
  for $s \in T$ **do**
    $W \leftarrow b_{\text{merge}}(s|\phi, D, G)$ ▷ localize goal in G
    if $W \neq \emptyset$ **then**
      $s \leftarrow$ nearest $w \in W$ in ϕ space
    else
      $V \leftarrow V \cup \{s\}$
    end if
    $E \leftarrow E \cup \{(s', s, 1)\}$
    $s' \leftarrow s$
  end for
  update ϕ
  update π, D
end for

We present an outline for online graph construction in Figure 1. If a state in a trajectory is similar to some state saved in the graph, then the corresponding nodes can be merged together. Formally, we build a directed, weighted graph $G(V, E)$ online as trajectories are stored in the replay buffer, as described in Algorithm 1. For each state s in the trajectory, we check if s is in the neighborhood of a state saved as a node in V. If multiple nodes satisfy the neighborhood check b, then we take the nearest state using the distance in the embedding space ϕ. Otherwise, if s is not localized to any states in V, then we add s to V. An edge is added to E that connects s with the node of the previously observed state s' of the trajectory.

C. Planning for exploration

By planning on the topological memory during training, the agent can set a sequence of intrinsic subgoals that act as a curriculum for reaching longer-horizon goals. In particular, the policy must succeed in reaching subgoals in the order given by the plan. Figure 2 presents a diagram of our method of planning to explore using the graph. We describe our formulation for open loop planning in Algorithm 2. First, the goal is localized in the graph using the neighborhood

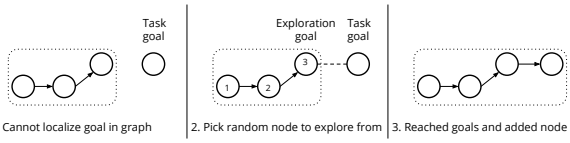


Fig. 2: Diagram of planning to explore using online topological memory. If the given goal cannot be localized in the graph, then a random state in the graph is selected as the goal. The agent attempts to returns to that state and then explores. If the task goal is successfully achieved, then a corresponding node is added to the graph.

Algorithm 2 Planning with Online Topological Memory

Require: timestep t, current state s, desired goal g, embedding ϕ, neighborhood function b, graph $G(V, E)$, value function D

if $t = 0$ **then** ▷ open loop planning
  $W \leftarrow b_{\text{merge}}(g|\phi, D, G)$ ▷ localize goal in G
  if $W \neq \emptyset$ **then**
    $g \leftarrow$ nearest $w \in W$ in ϕ space
  else
    select random $g \in V$
  end if
  subgoal plan $P \leftarrow ShortestPath(s, g, G)$
end if

$w_i \leftarrow$ current subgoal of plan P
if $w_i \in b_{\text{reached}}(s|\phi, D, G)$ **then**
  $w_i \leftarrow w_{i+1}$ ▷ get next subgoal
end if
$g \leftarrow w_i$

function b introduced in Section IV-A. If the desired goal cannot be localized in memory, then the planner selects a random node in the graph as the goal. Then, a shortest path between the localized start and goal nodes in the graph is computed using Dijkstra's algorithm. The learned goal-conditioned policy attempts to execute subgoals in this plan until the episode ends.

D. Goal relabeling

HER [8] relabels transition goals by selecting states encountered during a trajectory and by assigning a success reward (e.g. 0) instead of the prior reward (e.g. -1) if the new goal is achieved. However, HER assumes access to a function $reached(s, g)$ with domain covering the entire goal space that can check whether some goal is achieved in any given state. For goal-reaching tasks such as those with visual inputs, this assumption does not hold.

Instead, we propose goal relabeling using topological memory and the neighborhood function b. We can consider a goal g as reached if the current state s is within the neighborhood of g. In particular, s and g should be in the same neighborhood of some node in the graph.

V. Evaluations

We evaluate our method for online topological memory in three environments: `FourRoomsPointEnv` [3] with thinned walls [4], `FetchPush` [34], and a fixed `DoorKeyMiniGrid` environment [35]. These different environments span discrete and continuous control, with ground-truth state or pixel observations, which we use to answer the following questions:

A. Does online topological memory enable long-horizon reasoning?

For a long-horizon task, we evaluate in `FourRooms-PointEnv`, depicted in Figure 3. This environment has (x, y) coordinate observations and a continuous action space with noise added to actions. We fix the start and goal positions to be the centers of the bottom left and top right rooms, respectively. The maximum episode length is set to 200 timesteps; note that over 100 timesteps are needed to move from the start to goal position. We use DDPG [31] to train the goal-conditioned policy. We evaluate two different strategies for goal relabeling, standard HER which uses all future states as potential goals for relabeling, and a strategy which only considers states of the next h transitions.

We repeat each experiment with the same 3 random seeds. We plot each random seed as a transparent line and the average across the 3 random seeds as the solid line in Figure 4. Results demonstrate that planning plus goal relabeling with online topological memory consistently reaches the task goal, corresponding to DDPG+Planning+HER in Figure 4. In contrast, DDPG+HER cannot solve the task in all 3 seeds within 100,000 steps. Without goal relabeling, DDPG alone is unable to solve the long-horizon navigation task.

We visualize the trajectory taken by the policy trained using our method in Figure 5a. Without planning subgoals, the local policy is unable to solve the task. By planning with topological memory, the agent executes a sequence of subgoals, and is able to achieve final task goal. In Figure 5b, we visualize the graph generated by online topological memory. The "task success" nodes correspond to the subgoal states visited by the policy among all successful trajectories. We also show the "exploration" nodes that the agent had visited before finding and localizing the final task goal in the graph.

B. Does planning yield structured, efficient exploration?

In Figure 3, we compare the different exploration outcomes for policies evaluated in Section V-A on `FourRooms-PointEnv`. While DDPG+HER is able to consistently explore parts of the adjacent rooms compared to DDPG only, both fail to discover the goal by exploration using additive Gaussian noise. With our method of planning using online topological memory, if the desired goal cannot be localized within the graph, then the planner randomly selects a node in the graph and computes the shortest path to that node. Note that DDPG+Planning is unable to solve the task because the local policy cannot complete the subgoals generated by the planner. By using goal relabeling, the policy

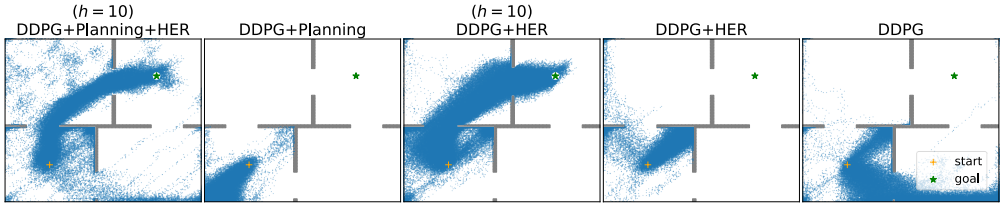


Fig. 3: Visualization of agent exploration, showing the points visited over the course of training on `FourRoomsPointEnv` (same seed).

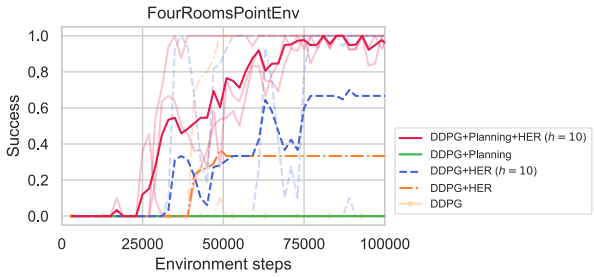
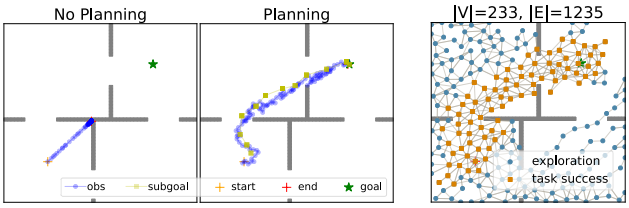


Fig. 4: Comparison between online topological memory (DDPG+Planning+HER) and different baselines, showing average task success evaluated through training on `FourRoomsPointEnv` (same 3 seeds). We plot each random seed as a transparent line and the average across the 3 random seeds as the solid line.



(a) Trajectory comparison (b) Graph

Fig. 5: (a) Comparison of trajectories with no planning and with planning in `FourRoomsPointEnv`. (b) Visualization of the graph constructed using our method.

learns goal-reaching behavior. Our method, corresponding to DDPG+Planning+HER in Figure 3, is able to consistently reach the desired goal in the diagonally opposite room. The same policy trained using our method was also visualized in Figure 5. By selecting a random state in the graph for exploration, the agent makes no prior assumptions on how to achieve the task goal, allowing it to escape the local minima of the corners of the room.

C. Can online topological memory handle multiple goals?

We evaluate whether our method can solve multi-goal tasks in `FetchPush`, where a robot must push a block to some given location on a table, indicated by the sphere in Figure 6. The goal position is randomly chosen as some point on the table for each episode. Results in Figure 7 show that online topological memory, labeled as

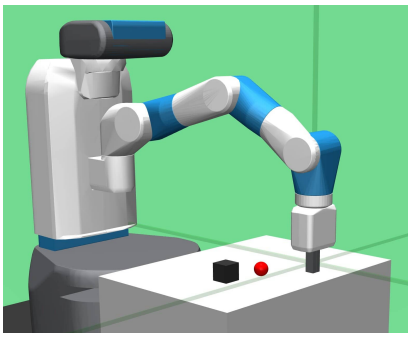


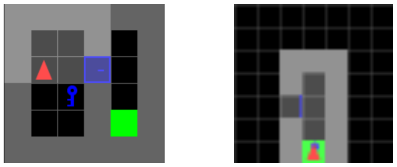
Fig. 6: Visualization of `FetchPush` environment, where the sphere indicates some goal position for the robot to move the block to. The sphere is for visualization only and not part of the environment.

Fig. 7: Comparison between online topological memory (DDPG+Planning+HER) and different baselines, showing average task success evaluated through training on `FourRoomsPointEnv` (same 3 seeds). We plot the average across the 3 random seeds as the solid line, and the shaded region corresponds to one standard deviation.

DDPG+Planning+HER, is able to achieve different task goals in fewer environment steps than DDPG+HER.

D. Can online topological memory perform visual planning?

We perform a simple evaluation on `DoorKeyMiniGrid`, an environment with a discrete action space and RGB observations. The agent must pick up a key, open a door, and navigate to the green goal tile. Note that while MiniGrid environments are randomly generated, for this evaluation we fix the environment to some seed, as shown in Figure 8a.



(a) Environment (b) Goal

Fig. 8: (a) Environment visualization for single DoorKey environment from MiniGrid. (b) Raw point-of-view RGB observation goal given to agent. The agent must match the position and direction specified by the goal. Note that though the door is open, the agent may re-close the door and still achieve the task goal.

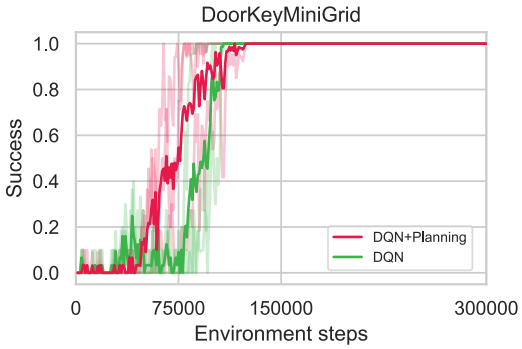


Fig. 9: Comparison of planning, showing average task success evaluated through training on `DoorKeyMiniGrid` (same 3 seeds). We plot each random seed as a transparent line and the average across the 3 random seeds as the solid line.

We give the agent a task goal specified by the RGB image shown in Figure 8b. For this simple visual setting, we use the l^2-norm in pixel space as the neighborhood function b. This will not scale to more challenging visual domains, and in future work we intend on learning a neighborhood function for pixel inputs.

We train DQN without goal relabeling, as a random exploration policy is sufficient for solving the task. For exploration, we use ϵ-greedy exploration that linearly decays ϵ from 1.0 to 0.001 over 200,000 environment steps. We chose to set $\epsilon_0 = 1.0$ to guarantee that the agent achieves the task goal. A more conservative exploration policy, such at $\epsilon_0 = 0.3$, could solve the task, but not consistently over different seeds. Due to RGB inputs, we learn a VAE representation; for more details on this choice, see Appendix VI-A.

In Figure 9, results show that our method (DQN+Planning) is more sample-efficient at learning. In Figure 10, we visualize an egocentric plan generated by our method, created by searching for paths in the graph.

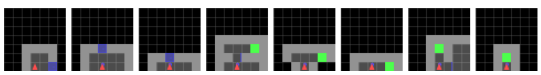


Fig. 10: Example of an egocentric plan generated by our method for `DoorKeyMiniGrid`.

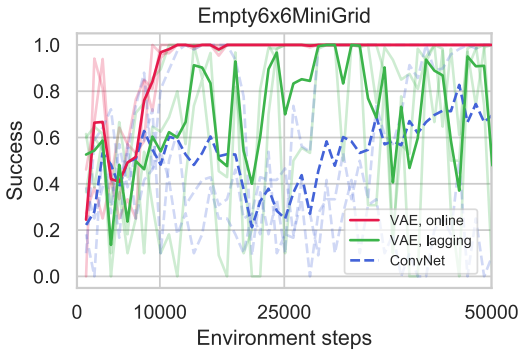
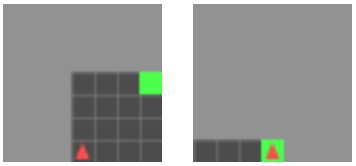


Fig. 11: Comparison between different visual representations, showing task success evaluated during training on `Empty6x6MiniGrid` (same 3 seeds). We plot each random seed as a transparent line and the average across the 3 random seeds as the solid line.



(a) Start (b) Goal

Fig. 12: Raw point-of-view RGB observations for Empty6x6 environment from MiniGrid.

VI. Conclusions & Future Work

We present online topological memory, an extension to prior work in visual navigation domains which construct graphs on observations for subgoal planning. Our method builds a graph on observations online. By dynamically constructing a graph, our method can plan subgoals while training the policy, guiding exploration and inducing a curriculum. In future work, we plan to evaluate our method on more challenging, multi-step continuous control tasks with image observations. For these domains, we will explore how to learn a neighborhood function online with pixel inputs for graph construction.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1659774. Eliot Xing is also supported by a CMU RI Lab Scholarship. Special thanks to Rachel Burcin and John Dolan for coordinating the Robotics Institute Summer Scholars program.

Appendix

A. Online VAE training

For MiniGrid environments with image-based observations, we learn a visual representation using a β-VAE. Here,

we compare a VAE representation against a representation learned by a standard convolutional neural network in a simple setting. The same layer architecture is used for both networks. We train DQN with an ϵ-greedy exploration policy that linearly decays ϵ from 0.3 to 0.001 over 37,500 environments steps. The embedding network is trained only on the agent observations and not on the task goal given to it, unless it has achieved that goal during some trajectory.

Results are shown in Figure 11 with the Empty6x6 environment from MiniGrid, where the agent must reach a fixed goal tile from the starting position, as shown in Figure 12. We plot each random seed as a transparent line and the average across the 3 random seeds as the solid line. Training a VAE representation online with the Q-function, the policy converges to the optimum. We also compare against a lagging VAE representation trained every 100 optimization steps of the Q-function. Policies learned with a lagging VAE representation or standard ConvNet fail to converge to the optimum.

References

[1] G. Matheron, N. Perrin, and O. Sigaud, "The problem with ddpg: understanding failures in deterministic environments with sparse rewards," *arXiv preprint arXiv:1911.11679*, 2019.

[2] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SygwwGbRW

[3] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 220–15 231.

[4] M. Laskin, S. Emmons, A. Jain, T. Kurutach, P. Abbeel, and D. Pathak, "Sparse graphical memory for robust planning," *arXiv preprint arXiv:2003.06417*, 2020.

[5] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*. Citeseer, 1993, pp. 1094–1099.

[6] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2011, pp. 761–768.

[7] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*, 2015, pp. 1312–1320.

[8] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[9] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Advances in Neural Information Processing Systems*, 2018, pp. 9191–9200.

[10] V. Pong*, S. Gu*, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep RL for model-based control," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Skw0n-W0Z

[11] D. Warde-Farley, T. V. de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih, "Unsupervised control through non-parametric discriminative rewards," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1eVMnA9K7

[12] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine, "Skew-fit: State-covering self-supervised reinforcement learning," *arXiv preprint arXiv:1903.03698*, 2019.

[13] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba, "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning," in *Proceedings of Machine Learning and Systems 2020*, 2020, pp. 9193–9204.

[14] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine, "Dynamical distance learning for semi-supervised and unsupervised skill discovery," in *International Conference on Learning Representations*, 2019.

[15] S. Venkattaramanujam, E. Crawford, T. Doan, and D. Precup, "Self-supervised learning of distance functions for goal-conditioned reinforcement learning," *arXiv preprint arXiv:1907.02998*, 2019.

[16] S. Nasiriany, V. Pong, S. Lin, and S. Levine, "Planning with goal-conditioned policies," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 814–14 825.

[17] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: a new approach for hard-exploration problems," *arXiv preprint arXiv:1901.10995*, 2019.

[18] ——, "First return then explore," *arXiv preprint arXiv:2004.12919*, 2020.

[19] H. Bharadhwaj, A. Garg, and F. Shkurti, "Leaf: Latent exploration along the frontier," *arXiv preprint arXiv:2005.10934*, 2020.

[20] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *International Conference on Machine Learning*, 2018, pp. 1515–1528.

[21] M. Fang, T. Zhou, Y. Du, L. Han, and Z. Zhang, "Curriculum-guided hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2019, pp. 12 623–12 634.

[22] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 875–12 884.

[23] Z. Deng, K. Narasimhan, and O. Russakovsky, "Evolving graphical planner: Contextual global planning for vision-and-language navigation," *arXiv preprint arXiv:2007.05655*, 2020.

[24] G. Zhu, Z. Lin, G. Yang, and C. Zhang, "Episodic reinforcement learning with associative memory," in *International Conference on Learning Representations*, 2019.

[25] Z.-H. Yin and W.-J. Li, "Toma: Topological map abstraction for reinforcement learning," *arXiv preprint arXiv:2005.06061*, 2020.

[26] G. Yang, A. Zhang, A. S. Morcos, J. Pineau, P. Abbeel, and R. Calandra, "Think, learn, and act on an episodic memory graph," in *BeTR-RL Workshop at ICLR*, 2020.

[27] W. Shang, A. Trott, S. Zheng, C. Xiong, and R. Socher, "Learning world graphs to accelerate hierarchical reinforcement learning," *arXiv preprint arXiv:1907.00664*, 2019.

[28] Z. Huang, F. Liu, and H. Su, "Mapping state space using landmarks for universal goal reaching," in *Advances in Neural Information Processing Systems*, 2019, pp. 1942–1952.

[29] N. Gothoskar, M. Lázaro-Gredilla, and D. George, "From proprioception to long-horizon planning in novel environments: A hierarchical rl model," *arXiv preprint arXiv:2006.06620*, 2020.

[30] T. Zhang, S. Guo, T. Tan, X. Hu, and F. Chen, "Generating adjacency-constrained subgoals in hierarchical reinforcement learning," *arXiv preprint arXiv:2006.11485*, 2020.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 449–458.

[34] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

[35] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.

Introspection in Theory of Mind Agents

Renos Zabounidis[1], Dana Hughes[2], and Katia Sycara[2]

***Abstract*— Theory of Mind (ToM) is a popular method for modeling human mental states. ToM models predict various aspects of mental states, including intentions, beliefs, emotions, and various other cognitive states. This mental representation can then be used to more effectively predict human actions, making Theory of Mind models widely applicable to any reinforcement learning task. ToM Models, however, are not without their down falls. Compounding errors, as well as rapid shifts in the cogitative states of a human, can cause the ToM model to fall out of sync with the players true cognitive state, causing action predictions to become worse then even mental state agnostic approaches. To remedy this problem, we propose introspection, an algorithm which updates the ToM model when it falls out of sync.**

***Index Terms*— Theory of Mind, Inverse Reinforcement Learning, Normalizing Flows**

I. Introduction

As autonomous, learning based systems rapidly become a part of every day life, it is important that they be able to complete their jobs in a socially aware context. There are many benefits for doing so. Socially aware agents are better suited to deal contextually with the needs of different people with different goals. In addition, a socially aware agent is more interpretable, being able to give the context of its beliefs on the humans mental state as justification for its actions at a particular point in time. However, endowing these systems with human like empathy and social skills is a nontrivial task. As humans, we regularly can rapidly infer other humans beliefs, desires, and intentions, using them to predict their future actions. In a computational framework, such inference would require knowledge of latent characteristics and states which are almost entirely inaccessible and computationally intractable [1].

Cognitive Psychology tells us that the brain does this by building layers of abstraction [2]. We do not, for example, attempt to simulate the individual neurons of others brains. Nor do we attempt to simulate any part of their brain at all, nor can we, since we never have direct access to the mind of another. Humans, and to lesser degrees other animals, posses a Theory of Mind, the ability to infer others observable actions in terms of unobservable mental states [3].

Creating machine agents with Theory of Mind has been tried in a variety of ways [3]–[5]. These approaches use hand crafted techniques spanning Inverse Reinforcement Learning, Bayesian Inference, and Game Theory [6]. In this work, we use an Inverse Reinforcement Learning framework as defined in [7] as a ToM model.

Inverse Reinforcement Learning (IRL) has classically only looked at determining stationary rewards [8]. In its extension to nonstrationary reward parameters, which bring the problem of drifting reward parameters and changing of intent, it becomes common for models to overtime become out of sync with the true reward parameters. The question is then how to resync the model. We postulate that the best we can do in such a case is to look at past experiences to make a somewhat reasonable estimate on where we should be. This 'introspection' on past experiences is done though modeling prior belief of the distribution of reward parameters. Incorporating variational inference, we can model the probability distribution over the set of reward parameters, and use this distribution in order to determine regions of high likelihood to reset our out of sync parameters.

We apply Introspection to a Grid World Search and Rescue scenario with various tasks with non-stationary reward parameters. We use the online IRL framework defined in [7], incorporating introspection where intent changes are detected. Introspection in this case is shown to improve reward parameter estimate during intent change detection.

II. Related Work

A. Inverse Reinforcement Learning

Much of Inverse Reinforcement Learning research focuses on finding better ways of identifying which reward function best explains a set of expert demonstrations. Bayesian IRL imposes and search's a prior distribution over reward functions [9]. Maximum Margin IRL chooses a reward function that most separates the optimal and second-best policy [10]. Gradient based approaches optimize a loss fuction with built in penalties, such the l2 norm of deviations from the expert's demonstrations [8]. Maximum entropy models attempt to directly maximize the entropy between the learned reward's policy and the original expert demonstration [11]. Our work is an improvement on the approach of [7], who use online, gradient based methods in order to estimate changing reward parameters.

B. Theory of Mind

Theory of Mind first originated in the field of Cognitive Psychology, where it was coined to refer to the cogitative capacity of an individual to attribute mental states to itself and others [12]. Since then, Theory of Mind has become a backbone explanation for a broad set of cognitive abilities from learning languages[28,29] to judging the guilt or innocence [30–32].

[1]Renos Zabounidis is with the College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, Massachusetts, USA rzabounidis@cs.umass.edu

[2]Dana Hughes and Karia Sycara are with the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA {danahugh,sycara}@andrew.cmu.edu

Theory of Mind was first applied to computational cognitive modeling thought understanding an agent's decision making using Baysian Inferance [3]. Computational Theory of Mind has been implimented using Deep Neural Networks [1], [13]. In addition, Inverse Reinforcement Learning Itself can be thought of as learning a ToM model by understanding the reward function learned to be the desires of the agent and the intentions to be the agents policy [14], [15].

C. Density Estimation using Variational Inference

Density estimation with generative is a well studied portion of statistical machine learning [16]. In recent times, Variational Inference based techniques have proven to scale well while providing state of the art performance. Such models include Auto regressive Flows [17], Variational Autoencoders [18], and Normalizing Flows [19]. In this paper, we make use Normalizing flows to preform maximum likelihood density estimation. Normalizing flows are used due to their property of having a closed form solution for likelihood.

III. Background

A. Reinforcement Learning

A Markov Decision Process (MDP) is defined as a tuple $\mathcal{M} = (S, A, T, r, \gamma)$ where,

1) S is a set of states
2) A is a set of actions
3) $T : S \times S' \times A \rightarrow [0, 1]$ is a transition function defining the probability of going from a state s to a state s' with action a.
4) $r : S \rightarrow \mathbb{R}$ is a reward function mapping a state to a reward.
5) $\gamma \in [0, 1]$ is a discount factor.

The goal of Reinforcement Learning is to learn a policy $\pi : S \rightarrow A$ such that the discounted sum of future rewards is maximized. To define what this means, let us introduce some notation. Firstly, define the value function conditioned on a policy as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} R(s_t)\gamma^t \mid s_0 = s, \pi\right] \tag{1}$$

We can then similarly define the value of a state action pair using the Q function:

$$Q^{\pi}(s, a) = \mathbb{E}\left[R(s) + \gamma \sum s' \in ST(s, a, s')V^{\pi}(s')\right] \tag{2}$$

The optimal values of these functions are then:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

The goal of Reinforcement Learning is therefore to find a policy $\pi^*(s, a)$ such that $\pi^*(s, a) = \underset{a}{argmax} Q^*(s, a)$.

B. Inverse Reinforcement Learning

In IRL, our MDP is modified to not have a reward function, and instead has a set of expert demonstrations. The goal of IRL then is to learn a reward function with the goal of recovering the original policy.

This problem as it stands is ill posed. Simply assigning every state a reward of 0 is a valid solution. We therefore treat the problem finding a reward function which maximizes the likelihood of the observed demonstrations:

$$R = \underset{R}{argmax} Pr(\rho \mid R)$$

By posing R to be the motivations of the agent we wish to model, inverse reinforcement learning models can be interpreted as ToM models [15].

C. Normalizing Flows

Normalizing Flows [19], [20] model arbitrary probability distributions by defining a series of bijective transformations from a base distribution. By denoting a base variable z and the transformed variable as x, x = f(z) where $z \sim \pi_z(z)$. The base density $\pi_z(z$ is typically a simple distribution like a unit Gaussian. Under this formulation, we can evaluate the transformation using the change of variable formula:

$$p_{\theta}(x) = p_{\theta}(z) \left|\frac{\partial x}{\partial z}\right|^{-1} \tag{3}$$

Much of Normalizing Flows research focuses on defining increasingly expressive and computationally inexpensive invertable layers [16]. One such layer is the Affine Coupling Layer [21]:

$$y_{1:\mathrm{d}} = x_{1:\mathrm{d}} \tag{4}$$

$$y_{\mathrm{d+1:D}} = x_{d+1:D} \odot exp(s(x_{1:d})) + t(x_{1:d}) \tag{5}$$

where s and t are functions from $\mathbb{R}^D \rightarrow \mathbb{R}^{D-d}$. The Jacobian of this transformation is:

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{\mathrm{d+1:D}}}{\partial x_{1:\mathrm{d}}^T} & diag(exp[s(s_{1:\mathrm{d}})]) \end{bmatrix} \tag{6}$$

Since this Jacobian is triangular, the determinant can be computing without computing the Jacobian of either s or t, meaning we can make s and t arbitrary neural networks (maintaining the aforementioned input and output dimensionality). In this paper, we use the coupling layers defined in [21], who alternate the coupling layers such that all inputs to the flow are changed. With a normalizing flow, we can preform density estimation on any reward parameters we are given. We do this by calculating the log likelihood of the input:

$$p_{\theta}(x) = \pi_z(f_{\theta}^{-1}(x)) \left|det\left(\frac{\partial f_{\theta}^{-1}}{\partial x}\right)\right| \tag{7}$$

By our previous formulation, f_{θ}^{-1} is easy to compute, thus making normalizing flows an excellent choice for density estimation.

IV. Approach

A. Online Belief Updates

We will be using the Online IRL framework by [7], for implementation details, please see their paper. A high level summary is described in Algorithm 1.

Algorithm 1 Reward Parameter Estimate Update

Initialize conjugate distribution parameters
Initialize trajectory buffer, $\tau = \{\sim\}$
Initialize set of intents, $\boldsymbol{\Theta} = \emptyset$
Set decay rate, λ;
Set change of intent threshold, ϵ_{intent}

repeat
  Observe current state and agent action, (s, a)
  Append (s, a) to τ
  if τ full **then**
    Calculate $\boldsymbol{\theta}_t$
    Calculate $\tilde{\boldsymbol{\theta}}$ (Algorithm 2 in [7])
    Update conjugate distribution parameters
    Set $\tau = \{\}$
    if $KL(\boldsymbol{\theta}_t \| \boldsymbol{\theta}_{t+k}) > \epsilon_{intent}$ **then**
      Calculate θ_{new} (Algorithm 2)
      Initialize conjugate distribution parameters;
    end if
  end if
until forever

B. Change of Intent

Change of intent is defined as the KL divergence between two online IRL updates. The KL divergence increasing beyond a certain predefined threshold activates the introspection module.

C. Introspection Module

Introspection is defined from the principle that when a ToM model is out of sync with the true beliefs, the model must look at previous experiences. Thus, in addition to our IRL assumption of being given a set of trajectories $\rho_1, \rho_2, ..., \rho_n \in \rho$, we additionally assume being given a set of reward parameters $r_1, r_2, ...$ sampled from the reward function. It is important to know we make no assumption as to the relationship between the set of reward parameters and trajectories, as that would reduce the IRL problem to regression. With this set of reward parameters, we train a Normalizing Flow which learns the distribution over possible reward parameters.

Using this flow, the problem becomes how to generate higher likelihood reward parameters from our original sample. To do this, we use gradient decent on the current reward parameter estimate at the time of the intent change. If we allowed all parameters to very, this update would yeild the same solution every time, making introspection useless. Therefore, we use a learning principle called Neuraths Ship [22], a principle which states that under uncertainty of a true causal structure, human learners do not try to learn the entire structure all at once. Instead, they incrementally update their belief model one step at a time. In our framework, this corresponds preforming gradient ascent on each reward parameter separately, keeping the others static. By choosing to update the reward parameter which most increases the likelihood of the current set of reward parameters, introspection can be viewed as an incremental process. In Algorithm 2, we see an exact specification of Neuraths based Introspection.

Algorithm 2 Neuraths Introspection

Train or Load Normalizing Flow F
Load x, the last 10 time steps of estimated reward parameters
let lr be the learning rate
let steps be the number of steps gradient ascent is preformed for
let n be the number of reward parameters
let likelihood be a list
let new_candidates be a list

for i =1,2,...,n **do**
  Let x_i be x with all parameters except the ith being frozen
  for steps timesteps **do**
    Preform gradient descent on x_i
    $x_i = x_i - lr * F'(x)$
  end for
end for
out_var = argmin(likelihood)
return likelihood[out_var], new_candidates[out_var]

V. Evaluation

A. Environment Specification

We use a 20x20 gridworld environment with 3 types of tasks: fire, triage, and supply. There are a total of 20 task throughout the map, and the reward of each tasks varies thoughout time. In addition, we define three meta parameters, which are set before each task.

$$\theta_{fire}(t) = \begin{cases} 1 + a * sin(\frac{\pi t}{200}) & 0 \leq t < 400 \\ -1 & 400 \leq t < 600 \\ 1 + b * \cos(\frac{\pi(t-600)}{200}) & 600 \leq 1000 \end{cases} \quad (8)$$

$$\theta_{triage}(t) = \begin{cases} -b * \cos(\frac{\pi(t-600)}{200}) & 0 \leq t < 400 \\ c * -1 & 400 \leq t < 600 \\ b * \cos(\frac{\pi(t-600)}{200}) & 600 \leq 1000 \end{cases} \quad (9)$$

$$\theta_{supply}(t) = \begin{cases} -2 & 0 \leq t < 400 \\ c * 2 & 400 \leq t < 600 \\ -2 & 600 \leq 1000 \end{cases} \quad (10)$$

where $a, b, c \in [0, 1]$ are sampled from a uniform distribution. As with [7], we use a linear reward function. We define the reward function as follows: $R(s; \theta) = \theta^T \phi(s)$. Figure 1 shows an example map.

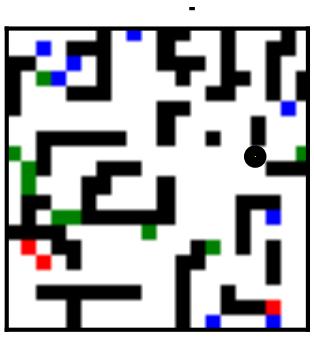


Fig. 1. An example map

B. Normalizing Flows Training

We implemented and trained the Normalizing Flow using the NuX library [23]. To build a dataset for training the flow, we sampled 200 values of each meta parameter. For each meta parameter, the resulting 1000×3 time series was split into 992 slices which were each 10 time steps long. These were reshaped to be a 30 dimensional vector, making a total of 198400 training points. The flow was then trained for 3000 batches. A batch size of 32 was used with the ADAM optimizer with an intial learning rate of $1e^{-3}$.

The flow itself consisted of 6 coupling layers, each of which had a 3 layer neural network with each hidden layer being fully connected and having 30 neurons. Figure 2 shows the training curve.

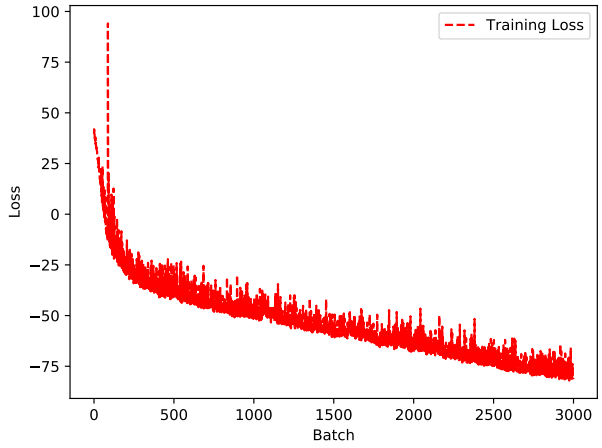


Fig. 2. Averaged reward parameter estimates though 50 trials

C. Results

Figure 3 shows the average reward parameter estimates through 50 trials. Results were not sigficantly better or worse then [7].

Table 1 shows the average squared error of the reward parameter estimates before and after the introspection update. Introspection is shown to significantly improve the reward estimate during intent changes.

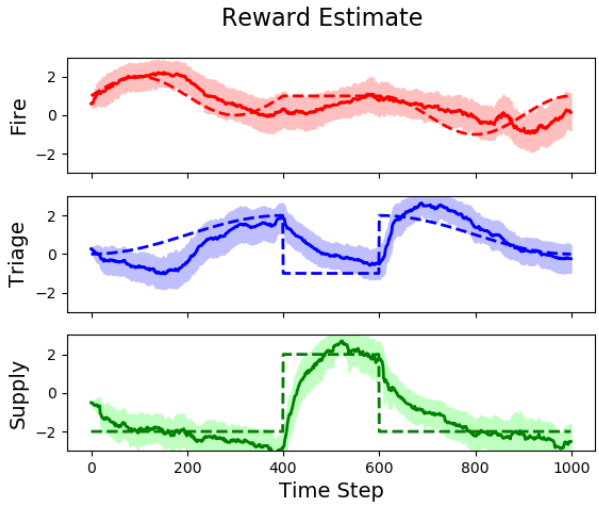


Fig. 3. Averaged reward parameter estimates though 50 trials

TABLE I
AVERAGE SQUARED ERROR AT INTENT CHANGES BETWEEN 50 TRIALS

	Squared Error
Before Introspection Update	8.52478692
After Introspection Update	6.49000124

VI. CONCLUSION & FUTURE WORK

In this paper, we create a method to preform introspection when a Theory of Mind (ToM) model is out of sync. This is done though modeling the distribution of possible reward parameters and incrementally updating the reward parameter which most increases the likelihood of the trajectory through gradient descent.

Our method was implemented in the Online IRL framework by [7] in a grid world search and rescue scenario.

Introspection ultimately fails in substantially improving the reward estimate. During intent changes, introspection overall lowers the L2 error between the reward estimate and the ground truth. This is promising in that it shows that introspection is an effective reset; however, the reward estimate is ultimately dominated by the psudoestimate over many time steps. This occurs due to the fact that the psudoestimate is calculated at every online IRL timestep independently, and therefore is not affected by the introspection update. This in turn means that even though Introspection can improve the reward parameter estimate, the pseudoestimate algorithm remains biased towards the previous estimate. This shows that the introspection module may be better suited for end to end approaches such as [1], where introspection may directly update ToM model parameters which affect future reward parameter estimates.

In the future, research may focus on better incorporating the changed reward estimate into future predictions. In addition, other Variational Baysian approaches such as Variational Autoencoders.

VII. Acknowledgment

The authors would like to thank Rachel Burcin, John Dolan, and all the other members of the RI community making RISS 2020 possible and their dedication throughout the summer. This work was supported by the CMU Robotics Institute Summer Scholars Lab Scholarship. This material is based upon work supported by the National Science Foundation under Grant No. 1659774.

References

[1] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine theory of mind," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 4218–4227. [Online]. Available: http://proceedings.mlr.press/v80/rabinowitz18a.html

[2] A. Gopnik and H. M. Wellman, "Why the child's theory of mind really is a theory," 1992.

[3] C. Baker, R. Saxe, and J. Tenenbaum, "Bayesian theory of mind: Modeling joint belief-desire attribution," in *Proceedings of the annual meeting of the cognitive science society*, vol. 33, no. 33, 2011.

[4] C. L. Baker, J. Jara-Ettinger, R. Saxe, and J. B. Tenenbaum, "Rational quantitative attribution of beliefs, desires and percepts in human mentalizing," *Nature Human Behaviour*, vol. 1, no. 4, pp. 1–10, 2017.

[5] J. B. Tenenbaum, "A bayesian framework for concept learning," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[6] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone, "Making friends on the fly: Cooperating with new teammates," *Artificial Intelligence*, October 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370216301266

[7] D. Hughes, A. Agarwal, Y. Guo1, and K. Sycara1, "Inferring non-stationary human preferences for human-agent teams," 2020.

[8] G. Neu and C. Szepesvári, "Apprenticeship learning using inverse reinforcement learning and gradient methods," *arXiv preprint arXiv:1206.5264*, 2012.

[9] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning."

[10] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning."

[11] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." 2008.

[12] A. I. Goldman *et al.*, "Theory of mind."

[13] T. Shu and Y. Tian, "M3rl: Mind-aware multi-agent management reinforcement learning," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[14] C. L. Baker, R. Saxe, and J. B. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 113, no. 3, pp. 329–349, 2009.

[15] J. Jara-Ettinger, "Theory of mind as inverse reinforcement learning," *Current Opinion in Behavioral Sciences*, vol. 29, pp. 105–110, 2019.

[16] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *arXiv preprint arXiv:1912.02762*, 2019.

[17] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems*, 2017, pp. 2338–2347.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[19] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.

[20] O. Rippel and R. P. Adams, "High-dimensional probability estimation with deep density models," *arXiv preprint arXiv:1302.5125*, 2013.

[21] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[22] N. R. Bramley, P. Dayan, T. L. Griffiths, and D. A. Lagnado, "Formalizing neurath's ship: Approximate algorithms for online causal learning." *Psychological review*, vol. 124, no. 3, p. 301, 2017.

[23] E. Cunningham, R. Zabounidis, A. Agrawal, I. Fiterau, and D. Sheldon, "Normalizing flows across dimensions," *arXiv preprint arXiv:2006.13070*, 2020.

[24] C. L. Baker, "Bayesian theory of mind: Modeling human reasoning about beliefs, desires, goals, and social relations," Ph.D. dissertation, Massachusetts Institute of Technology, 2012.

[25] J. Mendez, S. Shivkumar, and E. Eaton, "Lifelong inverse reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 4502–4513.

[26] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

riss.ri.cmu.edu