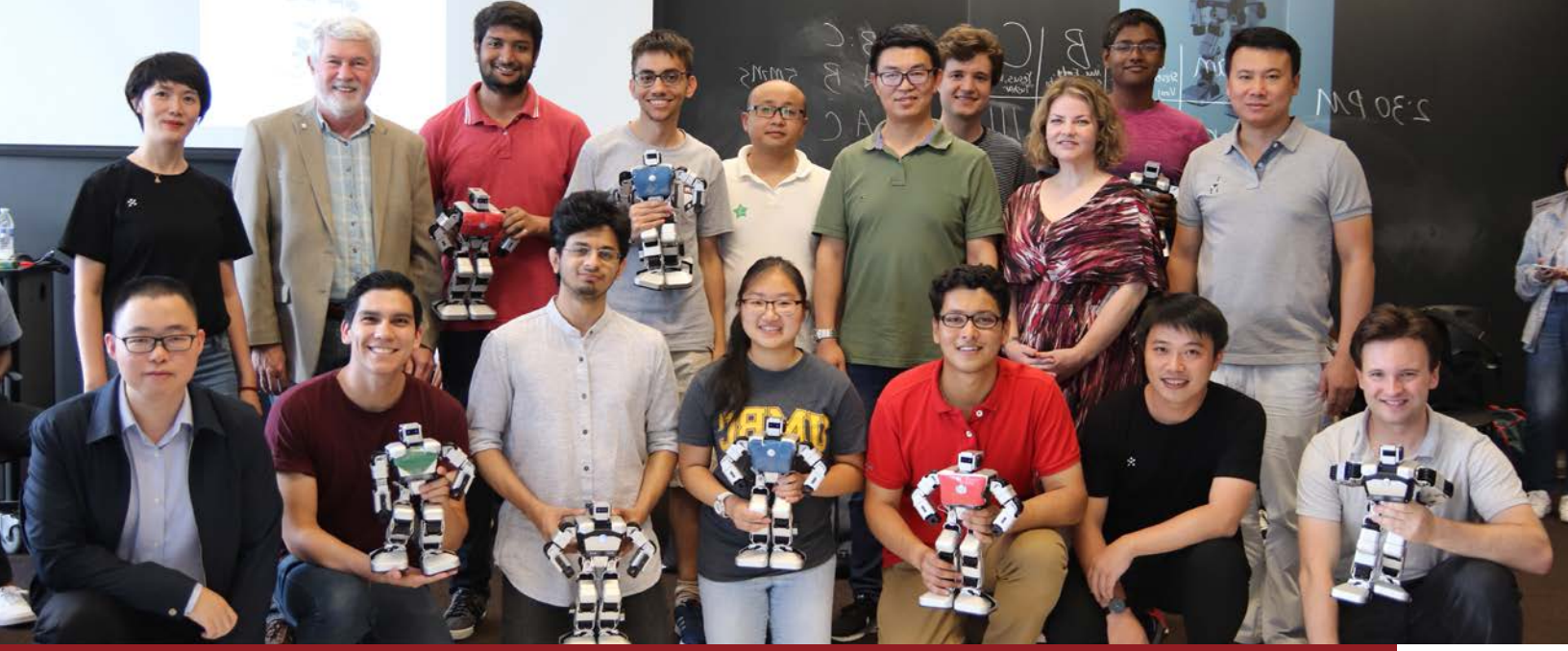
Carnegie Mellon University
The Robotics Institute

Working Papers **Journal**

Volume 6 | Fall 2018

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

RISS 2018 Journal Assistant Managing Editors

Stephanie Milani

David Russell

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.

Table of Contents

Table of Contents

Continued

About RISS

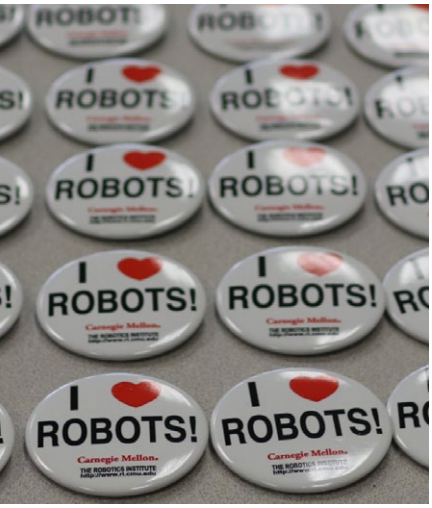
Robotics-related technologies are becoming ubiquitous and are dominating national headlines due to innovations such as driverless cars, service robots, surgical robots, and aerial vehicles. Robots and the knowledge required to create, operate, and interact with them will become increasingly important to society.

Launched in 2006, the Robotics Institute Summer Scholars (RISS) program is among the best and most comprehensive robotics research programs for undergraduates in the world. The RISS program immerses students in the world of robotics. Through RISS, students perform research under the mentorship of top scientists in robotics and intelligent systems at Carnegie Mellon University's Robotics Institute. The 50-plus participating mentors draw from a broad range of robotics research (e.g. field robotics, computer vision, machine learning, artificial intelligence, autonomy, graphics, human-robot interaction, and space robotics).

The RISS-guided research experience is coupled with powerful professional development in a nurturing global community and culminates in August with an annual research poster session and the publication of this RISS Working Papers Journal. The quality and breadth of research, high level of institute and university engagement, extensive professional development curriculum, graduate school application counseling, and alumni network create transformative experiences and remarkable post-program trajectories, with many students continuing to collaborate with CMU faculty members and the RI community.

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

Letter from the Editors



This journal is a forum for the member of the Robotics Institute Summer Scholars (RISS) program to share the results of their ten weeks of diverse robotics research with the broader community. To the best of our knowledge, it is the only journal for undergraduate students to document and publish their summer research in robotics. Preparing for and submitting to the journal adds significant value to the RISS program; over the course of the summer, developing their working papers journal submission helps Scholars focus on clearly-defined research questions. Developing this paper has also served as a starting point for conference and journal submissions: Scholars have a track record of acceptances at major venues for the work that they began during the RISS program.

Many people in the Carnegie Mellon Robotics Institute are responsible for the success of the Scholars. Throughout the program, the Scholars receive guidance and feedback from their faculty mentors, who are leaders in their respective fields. Most Scholars also are guided by graduate student mentors who assist them more with day-to-day tasks. In addition to helping them with the research, the faculty researchers and graduate mentors are often able to give highly targeted suggestions about the paper, given their deep knowledge of the field.

Much of the work that makes this journal possible is undertaken by the members of the RISS program. Scholars participate as peer reviewers, which improves the quality of papers and gives the students early hands-on experience with the review process. We would like to thank Yesus Yahel Becerril, Maggie Collier, Emily Duan, William Edwards, Kyuto Furutachi, Max Gordon, Mononito Goswami, Tessa Guengerich, Adarsh Karnati, Tushar Kusnur, Yiwei Lyu, Shiven Mian, Stephanie Milani, Viraj Parimi, David Russell, Scott Sussex and Brandon Trabucco for their contributions as peer reviewers. Max Gordon and Tushar Kusnur also deserve special recognition for assisting with edits, organization, and design.

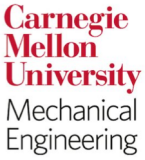
Additionally, we deeply appreciate Emily Ferris, Alex Hall, Dr. Juliann Reineke, and Dr. Joanna Wolfe from the Global Communications Center for reserving a significant number of appointments specifically for the journal, as well as holding events throughout the summer to help the scholars with the sometimes-daunting writing process. These critiques and workshops helped make the papers useful to researchers while still being relevant and accessible to the broader public.

Finally, we would like to thank Global Programs Manager Rachel Burcin and Principal Systems Scientist Dr. John M. Dolan for their tireless dedication to the RISS program and their guidance and support at every step of preparing the journal. In summary, this work would not have been possible without the effort and expertise of many people, and we are extremely grateful to each of them for their contributions.

Stephanie Milani & David Russell
Assistant Managing Editors

Thank you

At the core of the Robotics Institute Summer Scholars' program are incredibly talented and dedicated faculty, graduate students, staff, and RISS alumni. Their support, participation, leadership, and vision make this one of the best research experiences in robotics and intelligent systems in the world.



Government of Saudi Arabia

TOBITATE!

CMU Go Research Program

CMU Office of the Vice Provost for Graduate Education

An End-to-End Framework for Landslide Erosion Analysis

Hameed Abdul and Christoph Mertz

***Abstract*— We present a unified framework comprising numerous state of the art algorithms which improve both respond times to landslide events as well as intervention efficacy. This results in improved outcomes following landslide events including reduced property damage and saved lives. We discuss numerous code libraries and show that our system combines this information in a way which is intuitive to domain experts. Furthermore, we show that our framework can be implemented using inexpensive and accessible hardware, including a cell phone camera as well as a general purpose laptop.**

I. INTRODUCTION

Landslides are natural phenomena characterized by the downward slope movement of soil and rock, resulting in millions of dollars of damage including loss of property and loss of life. The frequency of landslides is increasing in Pittsburgh, due in part to record rainfall as well as a confluence of geological features which make the city susceptible to such seismic activity. The Landslide risk assessment and response process is accompanied by time-consuming multifaceted inspection. This includes drilling, geophysical studies, aerial reconnaissance, lab-based testing of earth materials and so forth [3]. Given that landslide site's are dynamic landscapes, traditional inspection methods are unable to quickly capture crucial land measurements that depict erosion or gradual change in the landscape.

Fig. 1. US Route 30 Landslide in Greater Pittsburgh Area

The Geospatial research community has made many contributions to solve this problem, the use of LIDAR or laser scanners and GPS markers to track the landslide site's kinematic displacement has achieved the best results in terms of accuracy [5, 4, 12].

However, LIDAR scanners and the accompanying equipment needed to capture both terrestrial and air-based scans in this method is inaccessibly expensive.

To reduce cost, studies on 3D reconstruction of landslide sites with high quality dslr camera equipped drones has been done with notable results [9, 14, 8, 6]. Although this method is far cheaper and only marginally less accurate than LIDAR based approaches, it is still costly and time consuming to systematically coordinate flight and image capture.

Furthermore, the image collection process is just the first step in the entire point cloud comparison procedure: images must be aggregated accordingly, an application must preform the 3D sparse then dense reconstruction (often different applications), a separate program must be used to visualize the results, numerous software applications must be used to align and segment point clouds for comparison and then finally point cloud distance can be computed. This method is inaccessible due to the technical knowledge and copious amounts of software needed to handle each step.

We aim to solve such problem, by creating an end-to-end framework that handles every step of 3D reconstruction and Geometric Change Detection with the only requirements being a camera equipped smart phone and a general purpose laptop. The following sections will give a brief overview of the each step in the pipeline and open-source software and algorithms used.

II. 3D RECONSTRUCTION

A. Structure From Motion

Structure From Motion(SFM) is a technique for estimating 3D scenes from a sequence of static 2D scene images coupled with local motion and is classically preformed in three general steps.

(1) Features are detected and extracted from each image by algorithms such as SIFT [2]. (2) Features are matched between images pairs, correlating features are saved while non-matching features are disregarded. If images are ordered (e.g. consecutive frames of a video) features only need to be matched with frames in close proximity. However, if images are unordered this process is exhaustive for it checks each image pair for scene overlap. (3) Geometric verification validates pairs after the feature matching process. Scene overlapping projective geometry is used to estimate transformations and 3D composition of detected features.

B. Multi-View Stereo

The output from SFM is used to compute depth and the surface normal information for every pixel in an image with overlapping pair. Overlapping images normal and depth

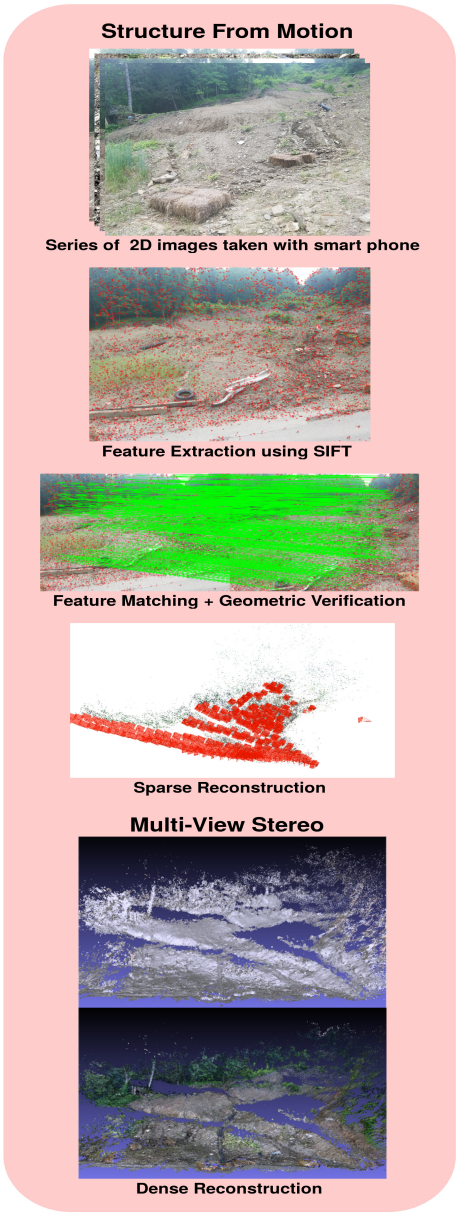


Fig. 2. 3D Reconstruction Process

maps are then fused to form a dense reconstruction. Poisson Surface Reconstruction and similar algorithms can reproduce 3D geometry of the scene.

III. Point Cloud Preparation

During the reconstruction process, the estimated camera position and orientation determine the initial transform of models. To properly compare models, the initially distance between clouds must be minimized. Methods of registration are used to reduce distance between transforms and align point clouds.

A. Local Registration

Given an initial transform and two point clouds, Iterative Closest Point Registration(ICP) roughly aligns the points [1]. However, ICP is local registration method and does not preform well if initialized transform do not converge. Colored Point Cloud Registration uses both point cloud geometry and color information and runs ICP iteratively with a joint optimization objective.

B. Global Registration

Local registration methods are not robust enough to handle point clouds with large initial distances or in congruent transforms. More sophisticated methods such as Fast Global[11] and Multiway[7] Registration handle initialization of transforms for non-converging point clouds rather well. Once global registrations provides an initialized transform, iterative local registration methods are once again applicable.

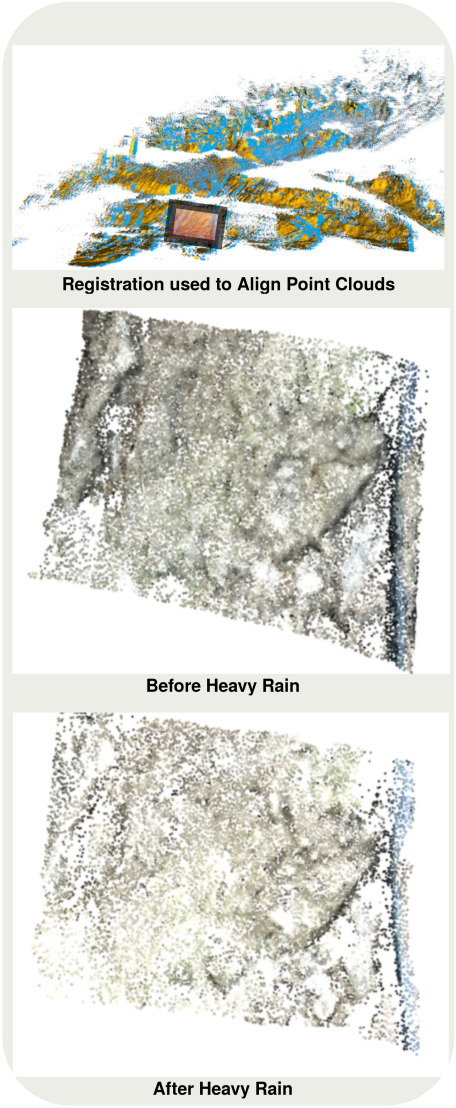


Fig. 3. Registration and Segmentation on two similar point clouds

IV. Calculated Point Cloud Distance and Change

Once the alignment of two point clouds are reduced, then the geometric comparison of two similarly structured

point clouds can be used to detect change that occurred. The Hausdorff distance is a straightforward solution for calculating the difference between aligned point clouds. This algorithm calculates the furthest corresponding point for each individual point and the resulting scalar values are then used to construct a heatmap to help visualize the change detected during comparison (see Fig. 4).

$$d_H(X, Y) = \max\{d(A, B), d(Y, X)\}$$

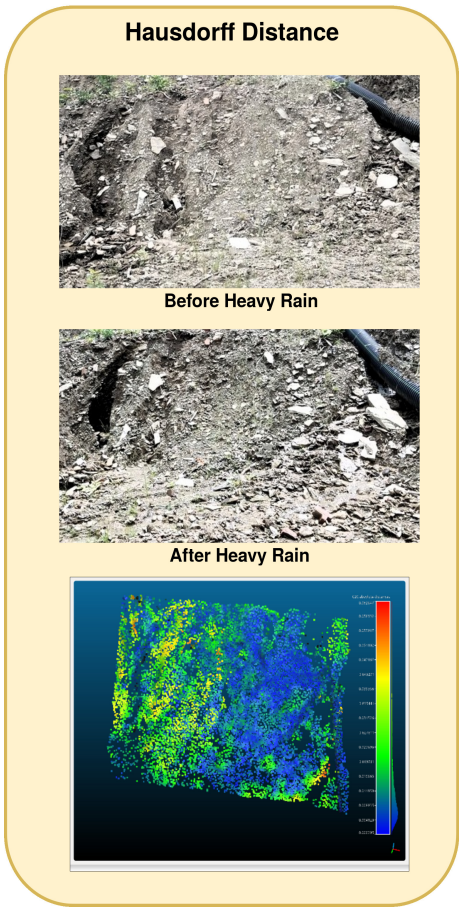


Fig. 4. Hausdorff Formula and Algorithm Heatmap

V. Framework Pipeline

(1) Image Collection. Images are captured from smart phone camera and scaled down to a 1/4 the resolution. (2) COLMAP 3D Reconstruction. The open-source tool COLMAP is used for the 3D reconstruction process [10]. It provides incremental reconstruction and bundle adjustment methods which provides state of the art performance compared to other implementations of SFM. (3) Registration and Segmentation. The Open3D library provides implementations state of the art registration algorithms as well as segmentation[13]. (4) Change Detection. Cloud Compare is used to turn the calculated scalar Hausdorff distance for each point cloud into a heatmap.

A. Limitations and Future Work

3D Reconstruction requires a well lit static scene. Vegetation, shaded areas, moving clouds, etc are obstructions and result in lost geometry in the reconstruction process. Therefor, this framework should only be used for sites with isolated or few areas of vegetation.

Currently, the use of the Hausdorff distance is straightforward and requires accurate elimination of outlier points in the segmentation process. Our next steps are to implement a method that better handles distance calculation between aligned point clouds with the presence of outliers.

Improving user interface, making use of Open3D's C++ interface to improve performance and further simplify the framework pipeline are areas we look to explore.

VI. Acknowledgment

This work was generously supported by Traffic21. Hameed would like to personally thank Tejas Khot and Wentao Yuan as well as the rest of the Navlab group for their advice, guidance and support.

VII. Conclusion

Our framework utilizes recent advances in open source libraries and algorithms to provide landslide responders with a single application that handles every step in the 3D reconstruction and change detection process.

References

[1] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM* (2001). ISSN: 15506185. DOI: 10.1109/IM.2001.924423.

[2] David G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* (2004). ISSN: 09205691. DOI: 10.1023/B:VISI.0000029664.99615.94. arXiv: 0112017 [cs].

[3] Lynn M Highland and Peter Bobrowsky. "The Landslide Handbook — A Guide to Understanding Landslides". In: *Landslides* (2008), p. 129. ISSN: 1067084X. DOI: Circular1325.

[4] Sajid Ghuffar et al. "Landslide displacement monitoring using 3D range flow on airborne and terrestrial LiDAR data". In: *Remote Sensing* (2013). ISSN: 20724292. DOI: 10.3390/rs5062720.

[5] D. Raucoules et al. "Time-variable 3D ground displacements from high-resolution synthetic aperture radar (SAR). application to La Valette landslide (South French Alps)". In: *Remote Sensing of Environment* (2013). ISSN: 00344257. DOI: 10.1016/j.rse.2013.08.006.

[6] Arko Lucieer, Steven M.de Jong, and Darren Turner. "Mapping landslide displacements using Structure from Motion (SfM) and image correlation of multi-temporal UAV photography". In: *Progress in Physical Geography* (2014). ISSN: 03091333. DOI: 10.1177/0309133313515293.

[7] Sungjoon Choi, Qian Yi Zhou, and Vladlen Koltun. “Robust reconstruction of indoor scenes”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2015. ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7299195.

[8] A. Stumpf et al. “Ground-based multi-view photogrammetry for the monitoring of landslide deformation and erosion”. In: *Geomorphology* (2015). ISSN: 0169555X. DOI: 10.1016/j.geomorph.2014.10.039.

[9] Abdulla Al-Rawabdeh et al. “Using an unmanned aerial vehicle-based digital imaging system to derive a 3D point cloud for landslide scarp recognition”. In: *Remote Sensing* (2016). ISSN: 20724292. DOI: 10.3390/rs8020095.

[10] Johannes L. Schonberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016, pp. 4104–4113. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.445. URL: http://ieeexplore.ieee.org/document/7780814/.

[11] Qian Yi Zhou, Jaesik Park, and Vladlen Koltun. “Fast global registration”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2016. ISBN: 9783319464749. DOI: 10.1007/978-3-319-46475-6_47. arXiv: 1311.2901.

[12] G. Bru et al. “Site scale modeling of slow-moving landslides, a 3d viscoplastic finite element modeling approach”. In: *Landslides* 15.2 (2018), pp. 257–272. ISSN: 16125118. DOI: 10.1007/s10346-017-0867-y.

[13] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: (2018). arXiv: 1801.09847.

[14] Sharad Kumar Gupta et al. “3D Reconstruction of a Landslide by Application of UAV & Structure from Motion”. In: (), pp. 1–7.

Design and Modeling of a Custom Unmanned Aerial Manipulator for Bridge Inspection

Nawaf Alotaibi[1] and Sebastian Scherer[2]

***Abstract*— Contact measurements are one of the key tasks performed during inspection of transportation infrastructure. Currently, these tasks require inspectors to manually climb onto the structure and record the measurements. In addition to the large cost of manual human inspection, these tasks introduce the risk of injuries to the inspectors due to working at high altitude. Current implementations for aerial manipulators inspection exist. However, these implementations lack stability and positioning accuracy for the end effector due to the external forces and the modification of the center of mass when the arm is carrying out a manipulation task. We propose to improve the efficiency of these measurements by developing an arm capable of contact attached to a custom tilted propeller UAV (Unmanned Aerial Vehicle). This paper introduces the technical details of the UAV and the arm design. We expect the arm to be a lightweight with weight distribution centered to the center of gravity to enable contact inspection and keep the system stable while performing its tasks. Moreover, this arm will facilitate the collection of coating thickness, structural integrity other data from the UAV using different contact sensors with modular implementation for the end effector.**

I. INTRODUCTION

Pennsylvania has 25000 public bridges. To conserve resources, a given bridge is only inspected once every two years, and inspectors rank the damage out of ten to gauge how much repair is needed [1]. While not highly common, there are documented instances of bridge inspectors being injured and/or killed on the job. These instances are particularly common during processes that require close proximity to the underside and support beams of a bridge [2][3]. This type of inspection entails an engineer climbing a bridge and using an ultrasonic sensor to detect the thickness of the gusset plates of a steel bridge [4]. The low frequency of inspection reduces the state governments ability to perform preventative maintenance on bridges that could save money in the long run and save the bridges from structural damage that results from the infrequent inspection.

One emerging field to deal with this problem is utilizing an Unmanned Aerial Vehicles (UAV) combined with the manipulation capabilities of robotic manipulators to achieve many different varieties of inspection, including contact bridge inspection. This combination of a UAV and a robotic manipulator forms an Unmanned Aerial manipulator (UAM). By using a UAM to complete inspection tasks, the process can be done in a way that is safer, cheaper, and more frequent than the current standard. Some Implementations of UAM for bridge inspection exist; however, they introduce a common trade-off of large size for the UAV to carry the weight of the manipulator, resulting in fewer areas to reach with the UAM. In addition, many UAVs implemented in current UAM have the known 4 Degrees of Freedom (Roll, Pitch, Yaw, and Altitude); however, these degrees of freedom are not enough to stabilize vehicle and the manipulator since any forces in lateral direction will result changing of the orientation of the UAM and shifting the manipulators from the point of inspection.

This paper proposes a solution to these issues by introducing a smaller UAM that is lightweight, able to carry similar payload compared to current UAVs implemented in this field, and complaint to forces in the lateral directions. The UAM consist of a custom tilted 6 Degrees of Freedom (DoF) multirotor UAV and a parallel linkage 3 DoF robotic arm. The tilted rotors UAV, shown in Fig. 1, generates forces in lateral directions, allowing the vehicle to move laterally without changing its orientation. This results in more stability to the arm and compensation for errors in roll and pitch. The diameter of the UAV is 690mm which is smaller than current UAV in this field. The arm design, shown in Fig. 2, is a parallelogram arm where the heavy components of the arm are at the base. This results in less shift for the center of mass of the UAM when the arm extends to do the inspection which adds more stability due to the lower moment from the arm when it is extended. The arm is powered using Dynamixel motors [5] that provide high torque and speed. Finally, rather than adding redundant motors for added maneuverability, the end effector, the part of the manipulator that interacts with the surface of the bridge, was outfitted with a passive 2 DoF using springs to allow it to flush to the surface of the bridge without adding excessive weight.

The UAM weights 4.5 kg with an additional payload of 2.5 kg. The vehicle was customized for large payloads so that no weight constraints will be limiting the design of the arm in worst case scenario or in case of adding more sensors to make other types of inspection. The combination of tilted multirotor small frame UAV and weight-centered robotic arm results in stable positioning for taking measurements and more reach for the UAM to collect data in hard to reach spots.

II. RELATED WORKS

Currently, contact inspection is performed by technicians who get readings from an ultrasonic thickness sensor [2]. Because this requires the sensor to contact the surface of the

[1]N. Alotaibi is with the Department of Mechanical Engineering at Georgia Institute of Technology Atlanta, GA 30332, USA nalotaibi8@gatech.edu

[2]S. Sherer is with the Faculty of the Robotics Institute, Department of Computer Science at Carnegie Mellon University, Pittsburgh, PA 15213, USA basti@andrew.cmu.edu

Fig. 1: Custom tilted hexarotor UAV with 6 DoF

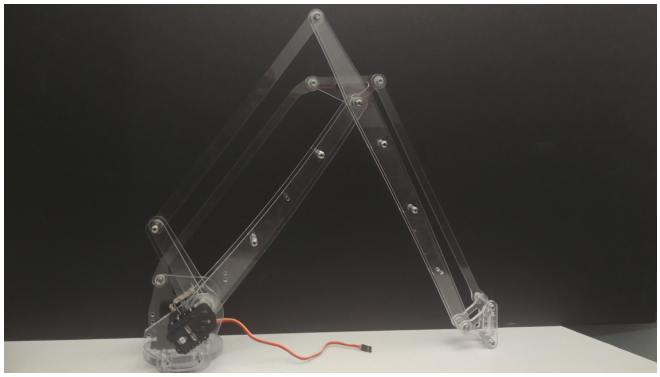


Fig. 2: Custom tilted hexarotor UAV with 6 DoF

bridge to collect data, a technician must climb the bridge or else use a cherry picker to access hard to reach areas. This can be a slow and dangerous process easily improved by making use of a UAV equipped with an ultrasonic sensor.

Several labs that specialize in UAV application have begun to explore the possibilities of UAMs. One such example is the aerial manipulator designed by T. Bartelds et al [6]. The UAV used in this design is a quadrotor with a 1 DoF prismatic arm. This particular model was designed primarily for the purpose of cushioning the impact between the UAV upon which it is mounted and a wall. The task is managed using an elastic in combination with a locking mechanism. The problem with this approach is that without the locking mechanism, the kinetic energy absorbed by the elastic results in oscillations that slow the process down and threaten to destabilize the UAV. When the locking mechanism is in place, the issue lies in the fact that after a single impact, the UAV must land so that the mechanism can be reset. In either event, time is wasted from one impact to the next. To avoid this, this manipulator is articulated, to allow for shock absorbency without undesirable oscillation. Rather than incorporating a locking mechanism that requires the drone to land and be reset manually, the manipulators articulated design allows it to cushion impacts repeatedly without stopping.

Another recent example is a UAM being developed is a human-size highly dexterous dual arm system introduced in [7][8]. Each arm is 5 DoF UAV mounted manipulator used for complex aerial maneuvers. Unlike the manipulator designed, this design is purposed to work in pairs. The weight of each arm is 1.8 kg, which requires a pair to be mounted on a hexarotor for additional power. The UAV used to carry this design is DJI Matrice 600 [9]. The complexity of this design is unnecessary for the task of bridge inspection. In addition, the UAV size for this task introduces a limitation of fewer areas to do the inspection task. As a result, this particular model is incompatible with the objectives of this project. Given that weight is the largest concern in drone flying, adding more than what is necessary is inadvisable. To allow for longer flight times with fewer battery changes, the manipulator is designed to be as light as possible.

Some UAMs are parallel manipulators, rather than serial manipulators, such as the Dexterous Aerial Manipulator designed by Mina Kamel et al [10]. A parallel manipulator is made of multiple actuated manipulators that connect to a single end effector, which allows for high rigidity and precision of movement [11].

The Aerobi project is a final example of a manipulator designed to be mounted onto a UAV [8][12]. In this case, there are a few different ideas utilized specifically for the purpose of bridge inspection. One model simply uses a phenomenon called the ceiling effect. This allows a UAV with a sensor mounted on the top of it to cling to the underside of the bridge and complete contact inspection that way. There are a few problems with this technique. One issue is that there is too much time between each instance of contact due to the UAV needing to align itself properly with the surface it must cling to. Another glaring issue is that this particular technique only works with flat, horizontal surfaces. The UAV is unable to inspect support beams or areas of the bridge that lie at an angle. This leaves a large portion of the bridge unattended to. Aerobi has also produced several different articulated manipulators to complete inspection tasks, but these also have their disadvantages. These manipulators have between 3 and 7 DoF. The models that have 5-7 DoF have actuated end effectors, which adds both excess weight and complexity to the system. Meanwhile, the Aerobi UAM that has only 3 DoF lacks the passive 2 DoF that is included in the manipulator designed and analyzed in this paper. Table I summarizes the design choices made in each work.

III. UAV PARAMETERS AND BUILD

To carry a big payload, there are many factors that can be controlled in building the UAV. The Main factor for the UAV to carry more payload is the thrust vector. Thrust vector in standard UAV can be increased by increasing the diameter of the propellers, increasing the KV rating of the motor, or by increasing the number rotors or propellers. The approach in this field for lower power consumption is by increasing the number of rotors to six or eight rotors to get a hexarotor or an octarotor. Standard multirotor platforms are underactuated due to all the thrusters being aligned. For a multirotor to move from one position to another, the vehicle has to command a torque to change the orientation of the overall thrust vector as seen in Fig. 3. As the vehicle

TABLE I: SUMMARY OF THE UAM IN THE REFERENCES USED IN THIS PAPER

Reference	Manipulator mechanism	Manipulator's DoF	UAV's number of rotors
[7]	prismatic	1	4
[8][9]	revolute links	5	8
[11]	parallel	1	4
[12]	-	-	4

gets larger, the ability to precisely control position and tracking trajectories decreases because of the coupling of position control to altitude control. This results in larger offset from the target point set for the vehicle when it follows a trajectory and stops at a specific waypoint. Another issue with Standard planer multirotor is the ability to hold position and orientation which are essential for bridge inspection task. Planar multirotor needs to fly at a non-level attitude to compensate for disturbances. For having the ability to interact with the environment, especially in the wind, conventional multirotors may not perform satisfactorily. Another issue with underactuated designs is that if they are to manipulate objects, the vehicle will not be able to counter many of the resulting reaction forces. A third design that has been researched is fixed rotors that have been rotated to point in such a way that produce some horizontal force components [13]. This provides added functionality in flying level in wind and hovering in a non-level orientation while not increasing complexity or additional parts. There is some loss in efficiency but the added tracking ability could outweigh this. For this task error compensation for roll and pitch were necessary abilities that the system should have.

The UAV that was built to carry the UAM is a 6 DoF tilted hexarotor. The requirements for the design is for the vehicle to have an endurance of 30 minutes and an additional payload of up to 2 kg. The build parameters of the UAV is shown in Table II. One of the goals in building the UAV is to get higher hover time in order to have more time for mutltiple inspections and doing other tests. The weight of the UAV is 4.5 kg. Using eCalc to estimate the hover time, The UAV can hover for 30 minutes and carry an additional payload of 2 kg for 5 minutes (with a depth of discharge of 100%). The large payload is used to decrease the weight limitations on the design of the arm. The UAV uses Tarot 680 frame with a diameter of 685mm. The motors used are Tiger Motors U3-700 KV. The battery picked was the 6S (22.2 V) 16000 mAh 15C Tattu. The control and perception in the vehicle are done with a Jetson TX1 offboard computer [14] and the force commands were sent from the computer to a pixhawk 4 autopilot where the pixhawk only does the mixing for the

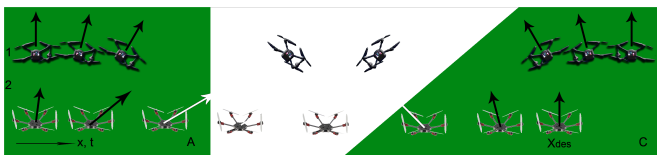


Fig. 3: Comparison between tilted multirotor and planer multirotor UAV going to point x_{des}

TABLE II: THE BUILD PARAMETERS OF THE UAV

Parameter	Value
Battery configuration	6S
Battery capacity	16000 mAh
Propeller diameter	13"
Propeller pitch	4.4"
K_v rating	700 RPM/V

motors. The pixhawk also sends IMU (100 Hz), barometer (25 Hz), magnetometer (25 Hz), and switch (25 Hz) data over USB. The camera used in the UAV is pointgrey blackfly that is facing forward. The camera outputs 320x240 images at 30 Hz. A laser rangefinder is also positioned at the front of the UAV. Using those two sensors, the distance from the walls can be obtained and the orientation of the UAV can be estimated from the visual data of the wall passed by the camera.

The motor tilt angle was set for the UAV to maintain its orientation with the external wind in consideration. Work was done to optimize the angles of motors to maximize the horizontal acceleration limits [13]. The maximum force used for calculating the tilt angle was based on a wind speed around 38 kph. Equation (1) shows the forces acting on the vehicle in the horizontal plane. Keeping the vehicle steady means to counteract the wind drag forces. Using (2) with an estimated area (S), drag parameter (C_D), and the speed mentioned, the needed motor tilt angle was found to be approximately 30 degrees.

$$m\ddot{x} = F_d + D \tag{1}$$

$$F_d = \frac{\rho v_{wind}^2 S C_D}{2} \tag{2}$$

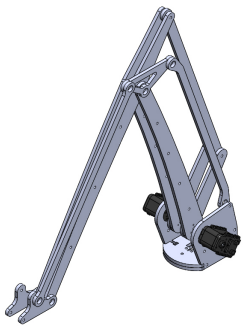


Fig. 4: CAD model of the manipulator

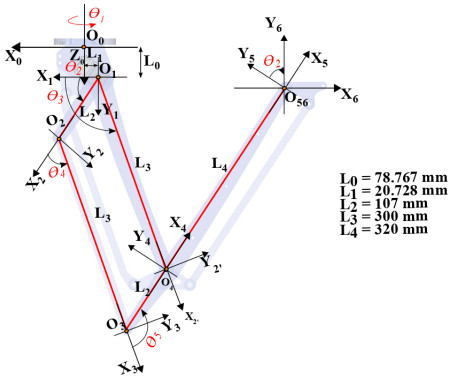


Fig. 5: The manipulator with dimensions and attached coordinate frames

TABLE III: DH parameters for the manipulator

Link	a_i	α_i	d_i	θ_i
$LinkO_0O_1$	$-L_1$	90^0	L_0	θ_1
$LinkO_1O_2$	L_2	0	0	θ_2
$LinkO_2O_3$	L_3	0	0	$\theta_3 - \theta_2$
$LinkO_3O_4$	L_2	0	0	$180 - \theta_3 + \theta_2$
$LinkO_1O_4$	L_3	0	0	θ_3
$LinkO_4O_{56}$	L_4	0	0	0

IV. MANIPULATOR DESIGN

The robotic manipulator used in the UAM is a 3 DoF parallelogram multi-linkage arm, shown in Fig. 4, based on the hobby robot arm uArm [15], an open source robotic desk arm and the ABB IRB460 robotic arm. The arm consists of links driven mainly by two motors at the base and a third motor that is used to rotate the whole arm. As a result, the center of mass is closer to the base of this. This is needed for the manipulator since a huge shift in the center of mass of the arm will cause a moment on the system that the UAV will have to compensate for. Another benefit of the parallelogram structure is the locked orientation of the end effector. The angle of the link driving the end effector is always equal to the link connected to the end effector. As a result, the end effectors orientation is always the same and the angle between the end effectors plane and the base plane is 90 degrees at any position the end effector is at. This property guarantees that the sensor will contact the wall at a normal angle without any change in the angle of contact that may result in the inability to take measurements.

A. Forward Kinematics

The manipulator system consists of a closed chain mechanism. The closed chain has only one loop in the shape of a parallelogram with two links connecting the loop the base and the other to the end effector. The forward kinematics model of the manipulator was calculated using the Denavit-Hartenberg (DH) convention. The coordinate systems and the geometric parameters used for calculating the DH parameters are shown in Fig. 5.

The DH parameters are presented in Table III. The cut joint that was used for the calculation is at O_4. Matrices of the homogenous transformations, according to the defined DH parameters are given by

$$A_1^0 = \begin{bmatrix} c_{\theta_1} & 0 & s_{\theta_1} & -L_1 c_{\theta_1} \\ s_{\theta_1} & 0 & -c_{\theta_1} & -L_1 s_{\theta_1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$A_4^1 = \begin{bmatrix} -c_{\theta_2} & s_{\theta_2} & 0 & L_3 c_{\theta_3} \\ -s_{\theta_2} & -c_{\theta_2} & 0 & L_3 s_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$A_5^4 = \begin{bmatrix} 1 & 0 & 0 & L_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The Frame O_6 is attached to link 2 whose orientation does not change during the motion. Its orientation is defined by the angle θ_2 regarding to frame O_5 according to

$$A_6^5 \begin{bmatrix} c_{\theta_2} & s_{\theta_2} & 0 & 0 \\ -s_{\theta_2} & c_{\theta_2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Matrix A_6^0 is

$$A_6^0 = A_1^0 A_4^1 A_5^4 A_6^5 =$$

$$\begin{bmatrix} -c_{\theta_1} & 0 & s_{\theta_1} & L_3 c_{\theta_1} c_{\theta_3} - L_4 c_{\theta_1} c_{\theta_2} - L_1 c_{\theta_1} \\ -s_{\theta_1} & 0 & -c_{\theta_2} & L_3 s_{\theta_1} c_{\theta_3} - L_4 s_{\theta_1} c_{\theta_2} - L_1 s_{\theta_1} \\ 0 & -1 & 0 & L_0 + L_3 s_{\theta_3} - L_4 s_{\theta_2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The position taken into consideration was the origin O_56 so the position of this point from the base is given according to the following

$$x_{pos} = L_3 c_{\theta_1} c_{\theta_3} - L_4 c_{\theta_1} c_{\theta_2} - L_1 c_{\theta_1} \tag{8}$$

$$y_{pos} = L_3 s_{\theta_1} c_{\theta_3} - L_4 s_{\theta_1} c_{\theta_2} - L_1 s_{\theta_1} \tag{9}$$

$$z_{pos} = L_0 + L_3 s_{\theta_3} - L_4 s_{\theta_2} \tag{10}$$

Equations 9,10, and 11 were implemented in MATLAB to plot the planer workspace at angle θ_1 equals 0 degrees. The planer workspace of the manipulator is shown in Fig. 6. According to the equations, the arm has a max reach in the x-direction of around 640mm. This provides an offset of approximately 200mm from the edge of the propellers to the origin O_{56}.

B. Compliant end effector

The design of the end effector is shown in Fig. 7. The end effector provides compliance by using spring to compensate for small perturbations in two DoF passive compliance. The end effector is also equipped with Force Sensing Resistors (FSRs) to give force feedback.

The Arm model weights around 700 g. Which is less than half of the payload of the UAV used in this system.

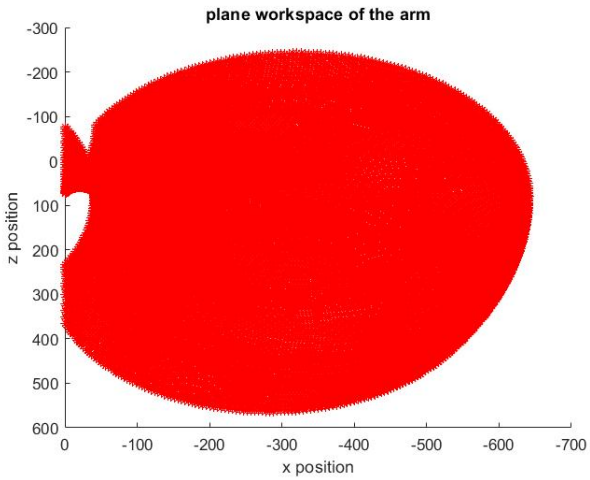


Fig. 6: Workspace of the manipulator

V. RESULTS

An experiment was set up to confirm if the UAV can respond to specific roll, pitch, and yaw command and passed on that follow these Command. The experiment was conducted by flying the UAV in altitude mode and sending roll, pitch, and yaw commands and based on that the error between the setpoints estimated by the drone and the commands sent can be estimated to check if the drone can fly accurately. Figure 8 shows the measurements taken from the flight test performed on the UAV. The UAV was able to follow the setpoints sent by the RC controller with a small margin of error with the maximum relative error in the roll and pitch around 6 degrees.

The vehicle was tested to maintain its orientation while moving laterally by implementing position control. The results of the test run is shown in Fig 9. The result shows that the vehicle can maintain its pitch and roll angles while moving in the lateral directions by producing forces in these direcitons. This result is important since it shows that the vehicle is able to maintain its orientation against external forces which increases the stability of the arm while doing the task. The shift of position was occurring because of the use of GPS which caused the vehicle to move a bit.

VI. CONCLUSIONS

This work demonstrated the design and modeling of a UAM that can be used for bridge inspection tasks using a 6 DoF tilted multirotor and a 3 DoF robotic arm with 2 additional passive DoF.

The UAV currently is able to fly like a planer UAV even with the tilted setup. The UAV can produce lateral forces to resist external forces and maintain its position. In the future, a controller will be implemented to give a robust position control. The controller should also be able to stabilize the UAV to a point which is possible using the lateral forces. For the arm, a control that depends on the state or the mission is needed for using different kinds of sensors or even different tasks like gripping or collecting samples. In addition, position

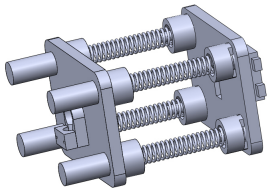


Fig. 7: passive compliance end effector

Fig. 8: roll, pitch , and yaw of the UAV sent by an RC

estimation will be implemented using a FLIR camera and optical flow point laser to give more accuracy instead of GPS.

From a design perspective, the manipulator in this work provides a large workspace and sufficient reach for the walls of the bridges. However, the end effector is not suitable for some parts of the bridge the surface is not flat but curved. A pulley-belt mechanism driven by a motor at the base can be used to control the angle of the plane of the end effector.

VII. ACKNOWLEDGEMENT

I would like to thank Dr. Sebastian Scherer for giving me the opportunity to learn and for his guidance and mentorship throughout the summer. I would also like to thank Dr. Oliver Kroemer, Sankalp Aurora, Azarakhsh Keipour, Lucas Nogueira, and other AirLab for their help. Huge thanks to

Fig. 9: Roll and pitch angle compared to x and y position of the UAV during test flight

Rachel Burcin and Dr. John Dolan for the huge effort that you put in the RISS program and giving us a chance to learn and improve. I also thank King Abdullah University for Science and Technology (KAUST) Gifted Students Program (KGSP) scholarship and the government of Suadi Arabia for their support and making this experience possible.

REFERENCES

[1] C. of Pennsylvania, Bridge information, Available at http://www.penndot.gov/ProjectAndPrograms/Bridges/Pages/default.aspx (2018/06/14).

[2] M. Hasch, Worker dies from injuries in east ohio street bridge accident, Available at http://triblive.co /news/adminpage/4224098-74/police-bridge-worker Accessed : 2018-07-23.

[3] B. Mayo, Bridge inspector fatally injured on interstate 279 on north side, Available at http://www.wtae.com/article/bridge-inspector-fatallyinjured- on-interstate-279-on-north-side/7461776 Accessed : 2018-07-23.

[4] M.P..S. Division, Bridge gusset plate ultrasonic inspection, Princeton Junction, NJ, Tech. Rep., 2009.

[5] https://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx Accessed: 2018-08-01

[6] S. H. S. S. T. Bartelds, A. Capra and M. Fumagalli, Compliant aerial manipulators: Toward a new generation of aerial robotic workers, IEEE Robotics and Automation Letters, vol. 1, no. 1, pp. 477483, 2016.

[7] A. F. Marco Tognon, Position tracking control for an aerial robot passively tethered to an independently moving platform, Rapport LAAS n 17107, 2017.

[8] A.Ollero, Workshop on autonomous structural monitoring and maintenance using aerial robots, 2017 International Conference on Robotics and Automation (ICRA), 2017.

[9] Matrice 600 specs, Available at https://www.dji.com/matrice600/info Accessed : 2018/08/01.

[10] K. A. M. Kamel and R. Siegwart, Design and modeling of dexterous aerial manipulator, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 48704876, 2018.

[11] E. by Serdar Kk, Serial and Parallel Robot Manipulators Kinematics, Dynamics, Control and Optimization. Rijeka, Croatia: InTech, 2012.

[12] Aerial robotic system for in-depth bridge inspection by contact." Available at http://www.aerobi.eu/about Accessed 2018/07/13.

[13] Jiang, G. and Voyles, R. (2014). A nonparallel hexrotor UAV with faster response to disturbances for precision position keeping. In Safety, Security, and Rescue Robotics (SSRR), 2014 IEEE International Symposium on, pages 15. IEEE.

[14] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/ accessed : 2018-08-01

[15] https://www.ufactory.cc/ Accessed 2018-08-01

Construction of a Map and a Human Driving Dataset from Birds-Eye View Video

Yesus Becerril[1], Zhiqian Qiao[2], and John M. Dolan[3]

[1]*Monterrey Institute of Technology and Higher Education*
[2,3]*Carnegie Mellon University*

***Abstract*— Machine learning approaches can generate better autonomous driving models and behaviors, but they need data. The only suitable public dataset for this purpose is the NGSIM dataset. However its size/time period and scope are limited. We therefore want to create our own datasets. Mounting fixed infrastructure requires permission, can be expensive, and is not portable. Using a drone gets around these problems. Creating such datasets requires creating a map that coupled with the recording and identification of the vehicles, allows us to obtain map and movement information. In this paper we introduce a method for obtaining such map information from aerial imagery.**

I. Introduction

Building machine learning models of drivers interactions with the environment, the road network and other traffic participants is a complex problem. To solve this problem it's needed trajectory data of vehicles as close as possible to reality which can also be a complex problem. For this reason, over the years the NGSIM program has been building datasets with information on the trajectory of the vehicles. These datasets has two main advantages. The first one is its high resolution, which allows researchers to investigate very detailed driving behaviors and to calibrate or estimate microscopic behavioral parameters and variables. The second one is its completeness in reflecting traffic conditions, which provides researchers with 100 percent and ground-truth traffic during the collection period and at the collection locations. Nevertheless, these datasets also has limitations:

- **Limitation-1** The time-space scope of the NGSIM trajectories is limited. For the freeway (US-101 and I-80) datasets, the time coverages are 45 min, and the space coverages are approximately 640 m and 500 m, respectively. For the arterial (Peachtree and Lankershim) datasets, the time coverages are 30 min, and the space coverages are approximately 640 m (five intersections) and 488 m (four intersections), respectively.
- **Limitation-2** The traffic conditions contained in the NGSIM datasets are limited. The traffic conditions contained in the freeway datasets are congested traffic conditions with oscillating features, except some pieces of high flow traffic exhibited by the exiting traffic in the US-101 dataset and the HOV-lane traffic in the I-80 dataset.
- **Limitation-3** The variety of data-collection roads is limited. The NGSIM datasets were collected only from freeways and arterials in U.S., making impossible to investigate the traffic on other level or type of roads, or on the roads in other countries.
- **Limitation-4** The variety of traffic components is limited. The NGISM datasets contain very limited trucks; for example, in the I-80 dataset, more than 80% vehicles are passenger vehicles with lengths smaller than 7 m. [4], Therefore, it is difficult to explicitly study on trucks behavior and impact by using the NGSIM datasets. [5]

Fig. 1: A digital video camera mounted on top of a building that overlooks a highway is recording vehicle trajectory data. [6]

In the other hand, there are previous works who search to solve these limitations, but have the following problems in the context of our application: they are time-consuming and costly, because some of them require expensive field surveying and labor-intensive post-processing.

Taking these limitations of the dataset and previous works into account, we decided to create a method that could achieve data collection with: Larger time-space scopes, various traffic conditions, different scenarios of roads and diverse vehicles types. Furthermore, a method that is cheaper, scalable, portable and easily repeatable, thus allowing to obtain more flexible datasets.

Creating such datasets requires the creation of a map. The flow of work is as follows, we first collect raw data from a drone, using a drone gets around several problems because of its low cost, hight flight altitude, stability and is ability to stay in the sky a long time. Then we process the video to get the vehicle detection. The result allows us to construct the map, in which by tracking the vehicle, we obtain map information such as movement information. We then perform trajectory smoothing, and finally, we will have our own dataset in NGSIM format.

This paper focuses on the construction of the map and the dataset. We have divided the process into 3 sequential stages:

- Extraction of the road from the image.
- Global Coordinate Road Geometry
- Transition to Local Lane Geometry

Through this discussion, we will show the process and evaluate the results of the construction of the map.

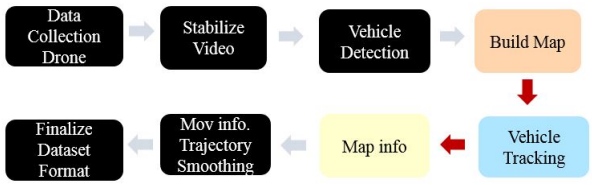


Fig. 2: Flow work of the entire method. Process covered in this paper, is highlight in color.

II. Extraction of the road from the image

The first step in the creation of the map is to extract only the area of interest, which in our case is the road network. We need to get rid of everything which is not road, because buildings, parks, trees, etc. Could represent noise.

To solve this problem, experiments were carried out on several recognized Convolutional Neural Network models for Semantic Segmentation, like fcn8, fcn32, [7] and segnet, [8], but in the end, we took advantage of the image segmentation network "U-net", described in Ronneberger et al. [1], with adjustments based on the work of Iglovikov et al, A. Buslaev and D. Gupta. [2] [9]

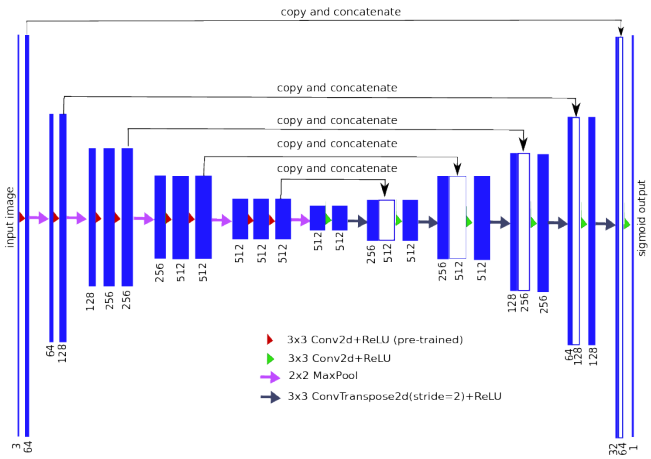


Fig. 3: Encoder-decoder neural network architecture, presented on Iglovikov et al. VGG11 without a fully connected layer is the encoder, and U-Net as the decoder.

Fig. 2 shows the model described in A. Buslaev in Iglovikov et al. There are many models based on VGG architecture, such as 11,13,16,19, depending on the number of layers in the model. In our work we decided to use VGG16 instead, as it's suggested on the words of the author. VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014, with a localization error of only 0.25. [10],

We preserve the decoder section of the network, because by adding a large number of feature channels, it allows the network to propagate context information to higher-resolution layers. In addition to the concatenations added after each up-sampling, this improves the localization and context.

Due to our small dataset, we couldn't train the network from scratch, for this reason the main change was in the encoder section, which was replaced by the down-sampling elements of the VGG16 in order to take advantage of the pre-trained weights on "ImageNet", which is an image database that contains almost 14,200,00 images. This allows us to have a base of features in our network.

TABLE I: Comparison between models, mean IoU

VGG-Unet Original	.65
VGG-Unet Reconstructed	***.87***
Unet without VGG16 weights	.42

The network was trained on a small dataset of 10 representative aerial road images, labeled in two classes. 0 for backgrounds and 1 for roads. The segmentation results are measured with its mean IoU, using a K-fold cross validation technique, the comparison of the different networks mean IoU results can be seen in table I. The results are divided into two parts. The first is based on the original result images. The second is based on the "reconstructed result" images, due to the fact that the results are not perfect and may have imperfections in them. The reconstruction consists of morphological techniques in which a structuring element or kernel, is slided through the image, similar to a 2D convolution, and depending on the neighboring pixels it will decide if that pixel is black or white. The technique is used to fill in the gaps in the image. This technique could give up to 20 percent of improvement, as it can be seen on figure 4 by using a small kernel of 5x5 for the erosion of the few incorrect pixels and a big kernel of 25x25 for the closing of the holes.

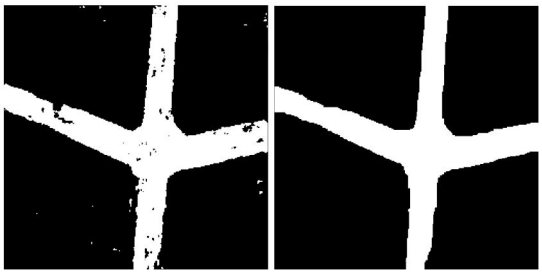


Fig. 4: Example of the improvement on the results. Original image (left), reconstructed (right)

III. Global Coordinate Road Geometry

A. Lines' Coordinates

Once we've get rid of the noise on the image (everything that is outside the road, such as buildings, trees, grass, etc.), we proceed to the identification of the several lane marks on the picture, from where we will build the map and obtain the reference points to build the global coordinates. This was achieved by using diverse methods of computer vision, such as Canny Edge Detection, Color Selection, Hough Transform Line Detection, Contours, etc.

Due to the different values needed in the multiple parameters to be used, we design a GUI to simplify the selection of values and increase the ease of repeatability. Where the algorithm is:

1) We apply a Gaussian Blur for the reduction of noise.
2) Depending on the color, we use HSV space color (Yellow tones), or HSL color space (White tones).
3) Then we have several morphological transformations where we can play with the structuring elements and the kernel size, to generate diverse alterations on the image that support our color detection.
4) Once we have the detected color, we apply another Gaussian filter to smooth out rough edges.
5) Finally, applying the method of Hough Lines or contours, we find the coordinate in pixels of the color on the image.

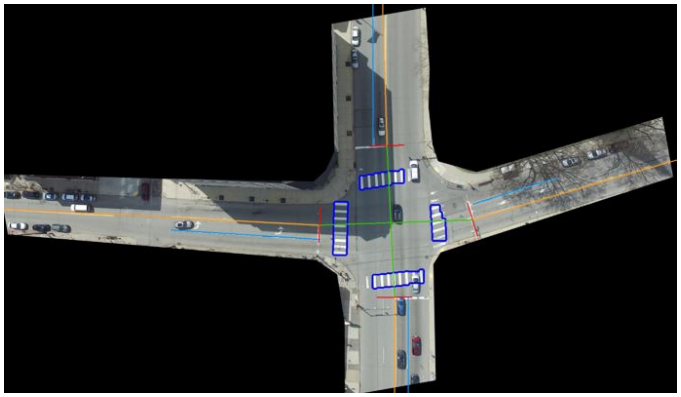


Fig. 5: We were capable of clearly distinguish each of the lane-marks on the road. Yellow line, White lines, Zebra crossing and the stop line.

B. Vehicles' Coordinates

Based on the locations of the Bounding Box achieved with RetinaNet, we identify the center location of the car at every frame of the video. But the obtained bounding boxes doesn't have the same orientation that the car has. As it can be seen on the image, the original bounding box, doesn't align respecting to the car, thus, in order to obtain better information of the width and length of the car in a future, we corrected the size of the bounding box, based on the angle of the street. With this information we will be able to calculate later more information about its position.

IV. Transition to Local Lane Geometry

A. Longitude and latitude coordinate of vehicles

Recapitulating, what we have until now are the coordinates of the lane marks and the center coordinates of the vehicles.

So, based on those values, we proceed to obtain the latitude and longitude from the center position of the car to the zero coordinate on the image, following the path of the yellow lane mark, which will be our axis X and Y.

At the end, we translate from pixel coordinate system to meters coordinate system by just multiplying our linear values with a constant value that represents the conversion of "pixels to meters". And finally, we are able to represent a map and a dataset, which contains the data of:

- Local X and Y
- Vehicle Size
- Section ID
- Lane ID

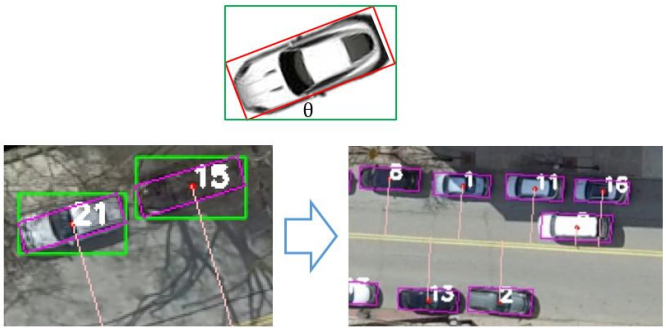


Fig. 6: Using the angle of the street, we determine the head angle of the car. At the end, we have the height and with of its center position.

V. CONCLUSIONS

Finally, we obtained a process capable of building a map, to obtain diverse values of the location of the driver about the street, in an x-y plane. One of the purpose of this method is to be scalable and easy to repeat, where our network has the ability to be trained on multiple scenarios, and the GUI makes the task of lanemarker identification easier. However, one next step could be to automatize the detection of the lanemarks as well as the road extraction, this could be made by using CNN for semantic segmentation. Moreover, working on create a bigger dataset of aerial road images, in order to increase the accuracy of our predictions. But, the most important next step for now, will be to tune this method on several scenarios in order to generate a more comprehensive dataset for modeling driver behavior.

ACKNOWLEDGMENT

I would like to thank Dr. John Dolan, and Zhiqian Qiao, for their guidance and support on this project. Furthermore, I'd like to thank Rachel Burcin, Dr. John Dolan and Ziqi Guo their efforts in organizing this amazing program.

Finally, thanks Tecnolgico de Monterrey for providing sponsorship and making this work possible.

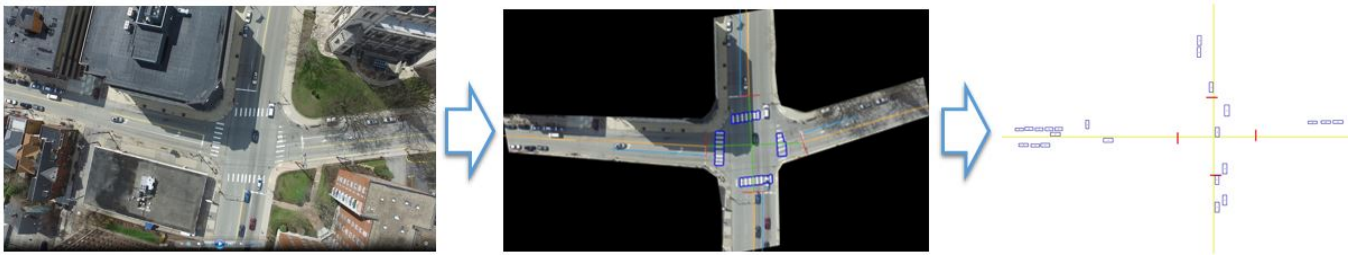


Fig. 7: Given an aerial video of an intersection, our method gets the mask of the road. Next, it locates the lane marks. Finally, we export the driver location values about those lane marks in an x-y plane.

REFERENCES

[1] O. Ronneberger, P. Fischer and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv:1505.04597, 2015.

[2] V. Iglovikov and A. Shvets, TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation, arXiv:1801.05746, 2018.

[3] H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.

[4] Wu, L.; Coifman, B. Improved vehicle classification from dual-loop detectors in congested traffic. Transportation Research Part C: Emerging Technologies 2014, 46, 222234.

[5] H. Zhengbing, Research based on high-fidelity NGSIM vehicle trajectory datasets: A review, DOI: 10.13140/RG.2.2.11429.60643, 2017.

[6] Office of Research, Development, and Technology, Office of Operations, RDT. (2017,08,09). Next Generation SIMulation Fact Sheet. [Online]. Available: https://www.fhwa.dot.gov/publications/research/operations/its/06135/index.cfm

[7] J. Long, E. Shelhamer, T. Darrel, Fully Convolutional Networks for Semantic Segmentation, arXiv:1411.4038v2, 2015.

[8] V. Badrinarayanan, A. Kendall, R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, arXiv: 1511.00561, 2015.

[9] D. Gupta (2018,04,12). Image-Sementation-Keras. [Online]. Available: https://github.com/divamgupta

[10] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556, 2014.

Homotopy-Based Footstep Planning for Humanoid Robots Operating in Complex 3D Spaces

Sahit Chintalapudi[1], Vinitha Ranganeni[2], Maxim Likhachev[3]

***Abstract*— Humanoid robots can be effective in a variety of domains from interacting with humans in the home to performing rescue operations. However, humanoid robots have many degrees of freedom which leads to a high dimensional state space that is computationally expensive to plan through. Previous work has studied how user-provided homotopy classes can be used to generate heuristic functions which guide footstep planning and alleviate this computational bottleneck. This approach, known as Homotopy-Based Shorted Path (HBSP) is limited in that it requires the 3D workspace to be projected into a 2D workspace before generating heuristic functions. Often robots need to be able to reason about stairs, ladders, ramps, and other features of the environment which a 2D projection cannot represent. In this work we propose an extension of HBSP that allows robots to generate heuristics from guidance given through higher dimensional workspaces. This is done by modifiying the traditional beam-graph approach to modeling topological constraintsto also capture movement between surfaces in the environment. We show that this approach allows us to extend HBSP to a variety of complex 3 dimensional environments.**

I. Introduction

Humanoid robots have been shown as an effective platform for performing a multitude of tasks in human-structured environments [1]. However, planning the motion of humanoid robots is a computationally-complex task due to the high dimensionality of the system. Thus, a common approach to efficiently compute paths in this high-dimensional space is to guide the search using footstep motions which induce a lower dimensional search space [2], [3]. This lower dimensional search space is a six-dimensional configuration space of a humanoid robot's feet consisting of x, y positions and orientation for each foot.

One approach that has been successful in footstep planning is using search-based planners such as A* [4], [5] and its anytime variants [3], [6]. To effectively plan footstep motions, these planners require heuristics to guide the search.

The Homotopy-Based Shortest Path (HBSP) algorithm automatically generates heuristic functions given a set of user-defined homotopy classes which are significantly easier to provide than hand crafted heuristics. [7] For each user-defined homotopy-class in the workspace, the algorithm generates a heuristic function by running a search from the goal to the start configuration while restricting the search to expand only vertices within the specified homotopy-classes.

These heuristics are then used in Multi-Heuristic A* (MHA*) [8] which is a recently-proposed method that attempts to leverage information from multiple heuristics. Roughly speaking, MHA* simultaneously runs multiple A*-like searches, one for each heuristic, and combines their different guiding powers in different stages of the search. Our work makes the assumption that there also exists a simple-to-define heuristic which is admissible and consistent.

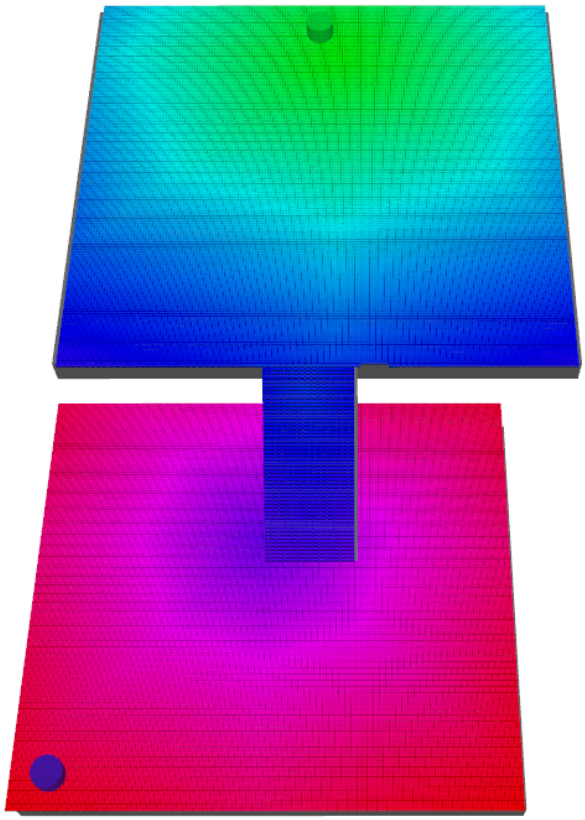


Fig. 1: an example heuristic generated by HBSP in a complex 3D environment

HBSP leverages a *pessimistic projection* of the environment from a 3D workspace to the 2D workspace. This pessimistic approach projects a cell (x, y, z) in a 3D workspace into a 2D workpace as an obstacle cell (x, y) if there is at least one z-value for which (x, y, z) is occupied. However, this approach does not work in an environment such as a multi-floored house because a pessimistic projection would no longer provide useful information. In this work, we investigate how to extend HBSP for such an environment (i.e

[1] Georgia Institute of Technology schintalapudi@gatech.edu
[2] University of Washington vinitha@cs.uw.edu
[3] Carnegie Mellon University maxim@cs.cmu.edu

a world that cannot be represented in a single 2D workspace).

Our work solves this problem by introducing what we call *gates* to our representation of the environment. Put simply, the introduction of gates allows multiple workspaces and records how a trajectory moves between them. Now, HBSP can guide the search in the direction of the transitions between workspaces demonstrated in the provided homotopies.

II. Related Works

A. Motion Planning Using Homotopy Classes

Homotopy classes have been frequently used to model the motion of a robot tethered to a fixed base point [9], [10], [11], [12]. The presence of obstacles introduces geometric and topological constrains for these robots. The constraints can create scenarios where the goal can only be reached if the cable configuration lies within a specific homotopy class, thereby making homotopy-based motion planning incredibly useful.

Homotopy classes have also been used in the context of human-robot interaction where a human wishes to restrict a robot's motion to specific homotopy classes [13]. For general approaches to explore and compute shortest paths in different homotopy classes, see [14], [15], [9].

B. Multi-Heuristic A* (MHA*)

The performance of heuristic search-based planners, such as A*, depends heavily on the quality of the heuristic function. For many domains, it is difficult to produce a single heuristic function that captures all the complexities of the environment. Furthermore, it is difficult to produce an admissible heuristic which is a necessary condition for providing guarantees on solution quality and completeness.

One approach to cope with these challenges is by using *multiple* heuristic functions. MHA* [16], [8] is one such approach that takes in a single admissible heuristic called the *anchor* heuristic, as well as multiple (possibly) inadmissible heuristics. It then simultaneously runs multiple A*-like searches, one associated with each heuristic, which allows to automatically combine the guiding powers of the different heuristics in different stages of the search.

Aine *et al.* [16], [8] describe two variants of MHA*, Independent Multi-Heuristic A* (IMHA*) and Shared Multi-Heuristic A* (SMHA*). Both of these variants gurantee completeness and provide bounds on suboptimality. In IMHA* each individual search runs independently of the other searches while in SMHA*, the best path to reach each state in the search space is shared among all searches. This allows each individual search to benefit from progress made by other searches. This also allows SMHA* to combine partial paths found by different searches which, in many cases, makes SMHA* more powerful than IMHA*. Therefore in this work we will use SMHA*. For brevity we will refer to SMHA* as MHA*.

III. Problem Definition

A. Notation

In order to extend HBSP to more complex environments, we begin by defining notation that allows us to capture these complexities. Let the robot's workspace be written $\mathcal{W}_3$. We assume that $\mathcal{W}_3$ can be represented as a series of flat surfaces $\mathbb{P}$; the i-th such surface is $\mathcal{P}_i$.

In the 3D environment there also exist obstacles given by the obstacle set $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, ...\mathcal{O}_m\} \subseteq \mathcal{W}_3$. For the sake of examining homotopy classes, we define a subset of the obstacles as *homotopy obstacles* $O^h \subseteq$. Considering only $\mathcal{O}^h$ in the planning process prevents us from over-constraining the heuristic around obstacles that aren't relevant to the final trajectory. The obstacles corresponding to plane $\mathcal{P}_i$ are given by $\mathcal{O}^i \subseteq \mathcal{O}$. We construct a family of 2D workspaces $\mathbb{W}$ by project the obstacles $\mathcal{O}^i$ onto $\mathcal{P}_i$ to generate the 2D workspace $\mathcal{W}_i^{2D}$. Fig. 2.

IV. Homotopy Classes of Curves

A. Homotopy Classes in a single 2D workspace

Informally, two continuous functions are called *homotopic* if one can be "continuously deformed" into the other. In general, uniquely identifying the homotopy class of a curve is non-trivial; however, if both curves are embedded in the plane, there exists a homotopy-invariant characteristic of the curve that can be computed.

In order to identify if two curves $\gamma_1, \gamma_2 \in \mathcal{W}_i^{2D} \setminus \mathcal{O}^h$ that share the same endpoints are homotopic we use the notion of h-*signature* (see [10], [9], [11]). The h-signature uniquely identifies the homotopy class of a curve. That is, γ_1 and γ_2 have identical reduced words if and only if they are homotopic.

In order to define the h-signature, we choose a point $p_k \in \mathcal{O}_k$ in each obstacle corresponding to that plane such that no two points share the same x-coordinate. We then extend a vertical ray or "beam" b_k towards $y = +\infty$ from p_k. Finally, we associate a letter t_k with beam b_k (See Fig. 3).

Now, given γ, let $b_{k_1}, \ldots, b_{k_m}$ be the sequence of m beams crossed when tracing γ from start to end. The *signature* of γ, denoted by $s(\gamma)$, is a sequence of m letters. If γ is intersected by the beam b_k, by crossing it from left to right (right to left), then the i'th letter is t_k ($\bar{t_k}$, respectively). The reduced word, denoted by $r(s(\gamma))$, is constructed by eliminating a pair of consecutive letters in the form of $t_k\bar{t_k}$ or $\bar{t_k}t_k$. The reduced word $r(s(\gamma))$ is a *homotopy invariant* for curves with fixed endpoints. It will be denoted as $h(\gamma) = r(s(\gamma))$ and called the h-*signature* of γ.

HBSP constructs paths which will be in the same homotopy class as a given reference path [7]. Thus, it will be useful to understand how the h-signature of a curve γ, which is a concatenation of two curves γ_1, γ_2 can be easily constructed. This reduced signature of $\gamma = \gamma_1 \cdot \gamma_2$ is simply the reduced signature of the concatenation of two curves' signatures $h(\gamma) = r(s(\gamma_1) \cdot s(\gamma_2))$.

B. Homotopy Classes across a series of 2D Workspaces

Our methodology is similar to the single workspace case when considering the family of workspaces $\mathbb{W}$. Beams allow us to record which obstacles a trajectory crosses and in what direction. We introduce a variation of beams called *gates* to

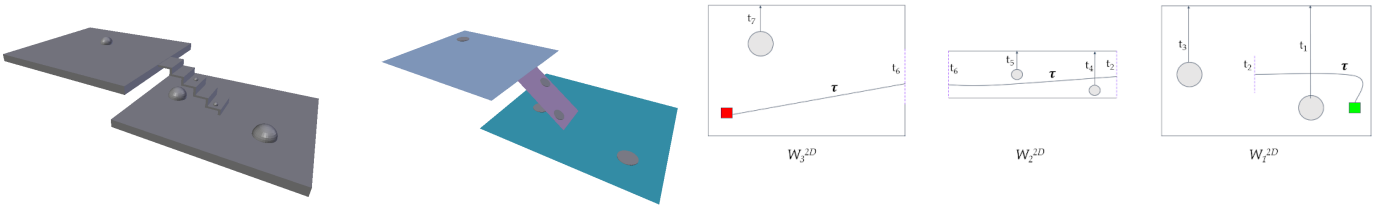


Fig. 2: (a) A 3D environment. (b) The set of flat planes where an obstacle will be projected on a surface. Circles show where on the surface an obstacle will be projected. (c) Visualization of $\mathcal{W}_1^{2D}, \mathcal{W}_2^{2D}, \mathcal{W}_3^{2D}$ The dashed magenta lines represent the gates.

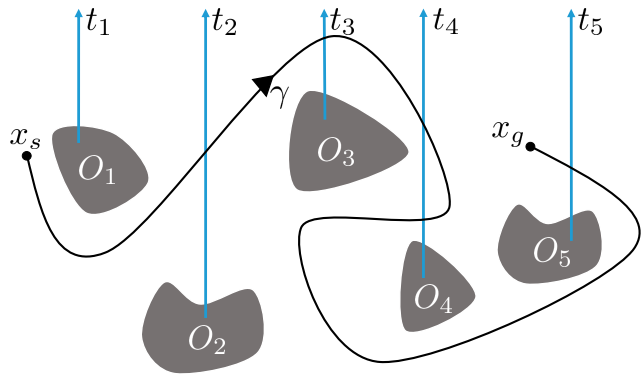


Fig. 3: The signature for this curve is $t_2 t_3 t_4 \bar{t}_4 \bar{t}_5$. The homotopy invariant of h-signature of curve γ is $t_2 t_3 \bar{t}_5$

record crossings between workspaces. A gate connecting $\mathcal{P}_i$ and $\mathcal{P}_j$ is given by $G_{ij} = \mathcal{W}_i^{2D} \cap \mathcal{W}_j^{2D}$ (i.e. the intersection between two 2D planes). We refer to the family of gates in the environment as $\mathbb{G}$. As is the case with beams, each gate has a corresponding letter t_k. Let the trajectory γ cross from $\mathcal{W}_i^{2D}$ to W_j^{2D}. If $i > j$ assign letter t_k. If $j < i$, assign letter $\bar{t}_k$.

V. Homotopy-Based Shortest Path

A. HBSP in a Single 2D workpace

We now describe the algorithm for computing homotopy-based shortest paths, or HBSP. Given a goal configuration q_{goal} and the graph $\mathcal{G}_{\mathcal{W}_2}$, HBSP incrementally constructs the augmented graph $\mathcal{G}^h_{\mathcal{W}_2}$ by running a variant of Dijkstra's algorithm from the vertex $(q_{\text{goal}}, \wedge)$. Here, $\wedge$ denotes the empty signature. For all vertices in $\mathcal{V}^h_{\mathcal{W}_2}$ that were constructed, the algorithm maintains a map $dist : \mathcal{V}^h_{\mathcal{W}_2} \to \mathcal{R}_{\geq 0}$ which captures the cost of the shortest path to reach vertices in $\mathcal{V}^h_{\mathcal{W}_2}$ from the vertex $(q_{\text{goal}}, \wedge)$. Given a vertex $(q_u, s_u) \in \mathcal{G}^h_{\mathcal{W}_2}$ and some user-defined signature s, this map $dist$ is used to compute the mapping d_s (which, in turn, is required to compute the heuristic function $\mathcal{H}_s$). Specifically,

$$d_s(q_u, s_u) = dist[q_u, h(s \cdot s_u)]. \tag{1}$$

Note that s corresponds to a signature of the path defined from the vertex $(q_{\text{goal}}, \wedge)$ towards the vertex (q_{start}, s), where

Algorithm 1 Homotopy-Based Shortest Path Algorithm

```
1:  function HBSP(Q, G_{W_2}, q_g, u, S)        ▷ u = (q_u, s_u)
2:      if Q = ∅ and dist[(q_g, ∧)] = NIL then
3:          dist[(q_g, ∧)] ← 0
4:          Q.add_with_priority((q_g, ∧), 0)
5:      while Q ≠ ∅ do
6:          v ← Q.extract_min()                 ▷ v = (q_v, s_v)
7:          if v = u then
8:              return (Q, dist[u])
9:          V_succ ← succ(v, S, G_{W_2})
10:         for v' ∈ V_succ do                  ▷ v' = (q'_v, s_v)
11:             alt ← dist[v] + length(v, v')
12:             if dist[v'] = NIL then
13:                 dist[v'] ← alt
14:                 Q.add_with_priority(v', dist[v'])
15:             else if alt < dist[v'] then
16:                 dist[v'] ← alt
17:                 Q.decrease_priority(v, dist[v'])
18:     return (Q, ∞)
```

q_{start} is a projection of the robot's start configuration. However, s_u is computed by the search as it progresses from $(q_{\text{start}}, \wedge)$ to (q_{goal}, s). Therefore $h(s \cdot s_u)$ corresponds to the remaining portion of the homotopy-based path specified by s after we remove its prefix that corresponds to s_u. For example, let $s = \bar{t}_3 \bar{t}_2 \bar{t}_1$ and $s_u = t_1 t_2$, then $h(s \cdot s_u) = h(\bar{t}_3 \bar{t}_2 \bar{t}_1 t_1 t_2) = \bar{t}_3$. Here, $\bar{t}_3$ is the signature of remaining portion of the path to the goal specified by s.

As the graph $\mathcal{G}^h_{\mathcal{W}_2}$ contains an infinite number of vertices, two immediate questions come to mind regarding this Dijkstra's-like search:

Q.1 When should the search be terminated?

Q.2 Should the search attempt to explore all of $\mathcal{G}^h_{\mathcal{W}_2}$?

We address **Q.1** by only executing the search if it is queried for a value of d_s which has not been computed. Thus, the algorithm is also given a vertex u and runs the search only until $d_s(u)$ is computed. This approach turns HBSP to an on-demand algorithm that produces results in a just-in-time fashion. It is important to note that when the search is terminated, its current state (namely its priority queue) is

Algorithm 2 HBSP Successor Function

```
1:  function succ(u, S, G_{W_2})                          ▷ u = (q_u, s_u)
2:      V_nbr ← neighbors(u)
3:      if S ≠ ∅ then
4:          𝕊 ← suffixes(S)
5:          for v ∈ V_nbr do                              ▷ v = (q_v, s_{u,v})
6:              if valid(v) then
7:                  if h(s_u · s_{u,v}) ∉ 𝕊 then
8:                      V_nbr.remove(v)
9:                  else
10:                     v = (q_v, s_u · s_{u,v})
11:             else
12:                 V_nbr.remove(v)
13:     return V_nbr
```

stored. When the algorithm continues its search, it is simply done from the last state encountered before it was previously terminated.

We address **Q.2** when computing the successors of a vertex described in Alg. 2 by restricting the vertices we expand. During the search, when we expand a vertex $u \in \mathcal{V}^h_{\mathcal{W}_2}$ in our Dijkstra-like search, we prune away all its neighbors $v \in \mathcal{V}^h_{\mathcal{W}_2}$ that have invalid signatures $h(s_u \cdot s_{u,v})$ (Alg. 2, lines 7-10). We define a valid signature $h(s_u \cdot s_{u,v})$ as one that is a *suffix* of a signature $s \in S$. Let $\mathbb{S}$ be the collection of all such signatures. That is, any signature $h(s_u \cdot s_{u,v})$, such that s could potentially be reached as the search progresses. More specifically, these suffixes identify the order in which certain beams can be crossed to reach a signature $s \in S$. For example, in Fig. 3, $S = \{t_2 t_3 \bar{t_5}\}$ and $\mathbb{S}$ of S is $\{t_2 t_3 \bar{t_5}, t_2 t_3, t_2, \wedge\}$. Here, $t_1 \notin \mathbb{S}$ as this beam does not need to be crossed to reach $t_2 t_3 \bar{t_5}$. Additionally, $t_3 t_2 \notin \mathbb{S}$ as the beams need to be crossed in the opposite order to obtain the signature $t_2 t_3 \bar{t_5}$.

The high-level description of our algorithm is captured in Alg. 1. The algorithm is identical to Dijkstra's algorithm[1] except that (i) when the cost $dist[u]$ is returned, the priority queue Q is also returned (Lines 7-8, 18) and (ii) the way the successors of an edge are computed (Line 9). Returning the queue Q allows the algorithm to be called in the future with Q in order to continue the search from the same state.

B. HBSP across a series of Workspaces

In this work we will need to construct a graph, $\mathcal{G}_{\mathbb{W}}$ embedded in the family of workspaces the robot operates in $\mathbb{W}$. Let every workpace $\mathcal{W}^{2D}_i \subseteq \mathbb{W}$ have a corresponding graph $\mathcal{G}_{\mathcal{W}^{2D}_i} = (V_{\mathcal{W}^{2D}_i}, E_{\mathcal{W}^{2D}_i})$. Then the vertex set is given by $V_{\mathbb{W}} = \bigcup_{i=1}^{n} V_{\mathcal{W}^{2D}_i}$. Similarly, the edge set is given by $E_{\mathbb{W}} = \bigcup_{i=1}^{n} E^{2D}_{\mathcal{W}_i}$

Let $B(\mathcal{O})$ be the set of beams associated with the obstacles in $\mathcal{O}$. Each of the beams in $B(\mathcal{O})$ have a unique letter associated with them. Each of the gates in $\mathbb{G}$ is also assigned a unique letter. Then the countably infinite signature set is given by $S(\mathcal{O})$ and it is defined as all the different h-signatures that can be constructed using the letters associated with $B(\mathcal{O})$ and $\mathbb{G}$. Now that we are equipped with the graph $\mathcal{G}_{\mathbb{W}} = (V_{\mathbb{W}}, E_{\mathbb{W}})$, we augment the graph with the signature set S which defines $\mathcal{G}^h_{\mathbb{W}}$. With this framework, we can call Alg. 1 with no modification.

[1]The only places where HBSP differs from Dijkstra's algorithm are the lines highlighted in blue.

VI. Acknowledgments

We are indebted to Fahad Islam and Andrew Dornbush for their advice and excellent mentorship over the course of the summer. We would also like to thank the Carnegie Mellon Univeristy Robotics Institute REU Site (NSF Grant #1659774) for funding this work.

References

[1] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*. Springer, 2014, vol. 101.

[2] K. Gochev, B. J. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path planning with adaptive dimensionality," in *SOCS Symposium on Combinatorial Search*, 2011.

[3] J. Garimort, A. Hornung, and M. Bennewitz, "Humanoid navigation with dynamic footstep plans," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3982–3987.

[4] J. E. Chestnutt, M. Lau, G. K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda ASIMO humanoid," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 629–634.

[5] J. E. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami, "An adaptive action model for legged navigation planning," in *IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 196–202.

[6] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 674–679.

[7] V. Ranganeni, O. Salzman, and M. Likhachev, "Effective footstep planning for humanoids using homotopy-class guidance," in *International Conference on Automated Planning and Scheduling*, 2018.

[8] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[9] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[10] D. Grigoriev and A. Slissenko, "Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane," in *International symposium on Symbolic and algebraic computation*, 1998, pp. 17–24.

[11] O. Salzman and D. Halperin, "Optimal motion planning for a tethered robot: Efficient preprocessing for fast shortest paths queries," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 4161–4166.

[12] S. Bhattacharya, S. Kim, H. K. Heidarsson, G. S. Sukhatme, and V. Kumar, "A topological approach to using cables to separate and manipulate sets of objects," *The International Journal of Robotics Research*, vol. 34, no. 6, pp. 799–815, 2015.

[13] D. Yi, M. A. Goodrich, and K. D. Seppi, "Homotopy-aware rrt*: Toward human-robot topological path-planning," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2016, pp. 279–286.

[14] S. Bhattacharya, "Search-based path planning with homotopy class constraints," in *AAAI Conference on Artificial Intelligence*, 2010.

[15] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, "Topological exploration of unknown and partially known environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3851–3858.

[16] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A," in *Robotics: Science and Systems*, 2014.

Eye Gaze Behavior during Teleoperation of a Robot in a Multi-stage Task

Maggie Collier[1], Reuben Aronson[2], and Henny Admoni[3]

***Abstract*— Individuals with motor disabilities can use assistive robotics to independently perform activities of daily living. Many of these activities, such as food preparation and dressing, are complex and contain numerous subtasks which can often be performed in a variety of sequences to successfully achieve a person's overall goal. Because different kinds of tasks require different types or levels of assistance, the variety of subtask sequences which the user can choose makes robotic assistance challenging to implement. However, a system that can anticipate the user's next intended subtask can optimize the assistance provided during a multi-stage task. Psychology research indicates that non-verbal communication, such as eye gaze, can provide clues about people's strategies and goals while they manipulate objects. In this preliminary work, we investigate the potential of using eye gaze for user goal anticipation during the teleoperation of a robot in a multi-stage task. We test our hypotheses by collecting eye gaze data during a user study in which participants teleoperate a 6-DOF robot arm to perform a food preparation task. Finally, we discuss our preliminary observations, such as the anticipatory gaze patterns which can be used for subtask anticipation.**

I. INTRODUCTION

A goal of assistive robotics is to enable individuals with disabilities to independently accomplish activities of daily living (ADLs). Many commercially available assistive robots, such as wheelchair-mounted robot arms [1], [2], are able to achieve that goal by taking direct input from the user through a controller interface (see Fig. 1). However, controlling a robot with numerous DOFs is extremely challenging because it requires a user interface that divides the DOFs into different lower-DOF modes. For example, the Kinova MICO arm's 6 DOFs can be separated into the following modes for a 2-DOF joystick: x-y mode, z-yaw mode, and pitch-roll mode. In this configuration, the user must press a button (located on top of the joystick in our experimental setup) to switch between modes while controlling the robot. Mode switching is a commonly used strategy for controlling a high-DOF assistive robot, but users of these commercially available systems express discontent with the strategy, citing the excess time and effort it takes to switch through modes [3]. To address this challenge, researchers have explored ways to add automation to these systems. An example of such a strategy is shared autonomy, in which an algorithm attempts to infer a user's goals from joystick inputs and assist in achieving them [4], [5].

[1]M. Collier is a Biomedical Engineering and Electrical Engineering student at the University of Alabama at Birmingham. maggieannecol@gmail.com

[2]R. Aronson is a Ph.D. student in Carnegie Mellon's Robotics Institute.

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

Fig. 1. Person wearing eye gaze tracker controls robot with joystick.

Shared autonomy has made improvements in the teleoperation of assistive robots. However, its success has only been demonstrated in simple tasks involving a single manipulation with an object [4], [5]. ADLs often do not resemble simple tasks: they are usually complex, representing multi-stage tasks with subtasks that require different interactions with different objects. One of many examples of an ADL that exhibits these characteristics is microwaving leftovers.

A person attempting to microwave leftovers might decompose the activity into the following subtasks: grasping the refrigerator door handle, opening the refrigerator door, retrieving a container with leftovers, transferring the container to a kitchen counter, taking the lid off the container, opening the microwave, transferring the container to the microwave, closing the microwave door, and starting the microwave. Imagine the difficulty trying to perform those subtasks with a teleoperated robot arm. Many of these subtasks require different manipulations from the end effector (grasp the door handle, push the button to open the microwave, move the container to the kitchen counter). This complexity is present in the teleoperation of many other ADLs and suggests that subtask-specific information is useful to assistive algorithms. Furthermore, enabling an assistive system to distinguish the current subtask and anticipate the next subtask during a multi-stage task would make it robust enough to switch to the appropriate assistance modes throughout any ADL.

While current assistive algorithms for teleoperated robots can predict a user's immediate goals, they are not equipped to anticipate a future goal [6] and, in effect, the next subtask during a multi-stage task. Psychology research in

hand-eye coordination indicates that eye gaze can provide insight into a person's strategies and future goals during object manipulation tasks. For example, Johansson et al. [7] reported that gaze precedes movement of the hand, and Pelz et al. [8] discovered occasional fixations to objects several seconds prior to people reaching for them during a hand-washing task. These behaviors provide information about future events, suggesting that gaze can be a powerful source of data for subtask anticipation. Aronson et al. [9] sought to determine if these same gaze behaviors are present when people attempt to manipulate objects with a teleoperated robot. In the study, participants were asked to spear a marshmallow with a fork that was secured in the robot's end effector. The results deviated from the findings in hand-eye coordination studies: most fixations that occurred during teleoperation were directed on the end effector of the robot, with very few fixations on the target object. These fixations on the end effector were used to track its location throughout the task. This monitoring is not present in hand-eye coordination studies because of human's proprioception: people know where their hands are without having to look at them.

Although the results in [9] indicate that gaze during teleoperation is largely used to monitor the end effector, we hypothesized that increasing the complexity of the task will cause more anticipatory gaze behavior to emerge. This hypothesis is based on Pelz et al.'s [8] work which suggests that complex, multi-stage tasks, such as ADLs, require people to strategize, making anticipatory gaze patterns more likely to occur. Thus, we aim to extend the work done in [9] by evaluating gaze behavior that results from the teleoperation of a robot in a multi-stage task.

In this work, we design a multi-stage task based on food preparation that has several subtask sequences which can lead to task completion, involves numerous objects, and requires several forms of object manipulation. We then develop a data collection study in which we collect eye gaze data as users attempt to perform the task with a teleoperated robot arm. Lastly, we discuss the preliminary observations from our study and the future directions of the work.

II. RELATED WORK

A. Hand-Eye Coordination in Psychology

An extensive body of work on hand-eye coordination exists in the field of psychology. Some early work on this subject focused on characterizing gaze behavior while people performed natural daily tasks, such as making tea [10], washing their hands [8], making a sandwich [11], [12], or driving [13]. This early work identified different gaze behaviors that provide insight into people's strategies while they perform natural tasks. For example, Pelz et al.'s [8] work, in which gaze was collected during a hand-washing task, identified "perceptual strategies," defined by the authors as behavior that seeks to optimize an overall task without aiding the immediate action in the task. As previously alluded to, the major perceptual strategy that their work defined is look-ahead fixations. Although not always referred to as such, gaze that preceded movement of the hand to an object within several seconds was reported by numerous other hand-eye coordination studies [7], [10]–[12], [14]. Hayhoe et al. [12] delved into the purposes of these planning behaviors and found that spatial orientations of task-relevant objects in a scene are stored in memory through multiple fixations, suggesting that look-ahead fixations are meant to more precisely locate an object to be manipulated in the near future.

While eye gaze reveals people's planning behavior, studies have shown that the fixations that occur while a person is performing a task are largely task-specific, meaning that people fixate primarily on objects that are related to the task at hand [10], [12], [15]–[18]. Because gaze tends to not deviate from features relevant to a task, our aim to use gaze to anticipate the progression of a multi-stage task seems feasible.

B. Eye Gaze in Human Robot Interaction

While eye gaze is commonly used in social robotics [19], less research has been conducted on using human eye gaze to predict human intent while a person performs a task. Of those studies, eye gaze is primarily studied in or used for a human-robot collaboration task [20]–[22]. Some of this work even uses eye gaze to determine an immediate subtask or anticipate a future subtask during a multi-stage task [20], [21], [23]. All of these studies differ from our work in that the users performed tasks with their hands rather than teleoperating a robot to perform the task. In this sense, our work is more focused on using eye gaze for assistive robotics, in which teleoperation is widely used in commercially available systems.

In the context of teleoperated robots, Admoni et al. [6] first described the potential of incorporating eye gaze into the state-of-the-art shared autonomy system described in [4] and [5]. The work in [6] inspired Aronson et al. [9] to explore eye gaze behavior during teleoperation of an assistive robot—the study that this work is extending. A recent follow-up paper from that work discussed the potential of using eye gaze for error detection during teleoperation [24].

Another notable study on eye gaze for assistive robotics is [25], in which an anticipatory control algorithm that uses gaze was presented and evaluated. Similar to [9] and this work, Huang et al. [25] was inspired by the look-ahead fixations reported in psychology research [8], [14]. In the study, a list of smoothie ingredients with their pictures was set before participants, and participants were asked to verbally express a desired ingredient to a robot. The anticipatory control algorithm aimed to use the participants' look-ahead fixations to anticipate their desired ingredient before receiving the verbal request. Our work differs from [25] in that it evaluates gaze behavior while a robot is teleoperated, since teleoperation is a more commonly used control method in commercially available assistive robots than verbal cues. Additionally, Aronson et al. [9] found that people's gaze patterns are significantly different when watching a robot move autonomously vs. via teleoperation:

behavior like monitoring glances are much more frequent in the case of teleoperation.

III. DATA COLLECTION STUDY

To investigate eye gaze behavior during the teleoperation of a robot in an ADL, we conducted a user study in which participants teleoperated a robot arm to perform a multi-stage task involving food preparation.

A. Design

To maximize the possibility of anticipatory eye gaze behavior, we created a multi-stage task that met the following criteria:

- More than one form of object manipulation is required to complete the task.
- More than one possible sequence of subtasks can lead to overall task completion.

Because food preparation is an ADL that often involves a variety of interactions with numerous objects, we designed a multi-stage task that required participants to serve a snack from objects in an open cabinet using a Kinova JACO robot arm. The task involved scooping spoonfuls of food from a snack container onto a plate. Fig. 2 shows the experimental environment and a zoomed-in image of the cabinet, in which the plate and spoon gripper were placed on the top shelf and the snacks on the bottom shelf. The position of each snack remained the same across trials and participants, but the position of the plate and spoon gripper were counterbalanced across participants. (The spoon gripper was placed to the left of the plate for some of the participants.)

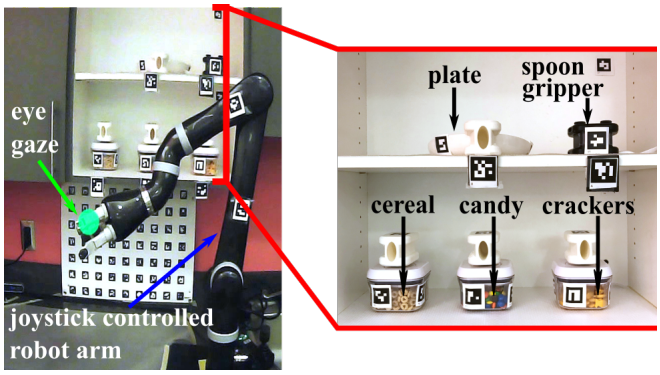


Fig. 2. The experimental environment and the open cabinet.

As shown in Fig. 3 and Fig. 4, grippers for the end effector were manufactured and glued to the dishes and snack containers to make grasping the objects easier. Although grasping the objects and scooping the food were two manipulations already required to complete the task, we added another required manipulation by purchasing snack containers that are opened by pushing on the lid, which presses down a button to disengage a vacuum seal (see Fig. 4). These containers are designed for individuals with disabilities.

In summary, we had 3 different required manipulations (grasp, push, and scoop) and 5 different objects. Because each trial only required scooping one snack, a minimum of 3 different objects had to be manipulated for task completion.

Fig. 3. The spoon gripper and assistive plate.

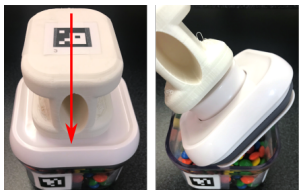


Fig. 4. Opening the snack containers.

Subtask sequence variation was introduced by allowing the participants to retrieve and manipulate objects in whatever order they desired. As an example, one participant might retrieve the plate, retrieve the snack container, retrieve the spoon gripper, and open the container before attempting to scoop the food, but another participant might retrieve the spoon gripper, retrieve the snack container, open the snack container, and retrieve the plate before scooping.

Throughout trials, gaze was collected using Pupil Labs Pupil [26], a wearable eye tracker with an egocentric (world view) camera and two cameras to record the individual eyes. The cameras tracking the eyes and the egocentric camera collected data at 120 Hz and 30 Hz, respectively. Fig. 1 shows a participant wearing the gaze tracker and the participant's position with respect to the robot's workspace.

B. Procedure

Each participant was given 5 minutes to acclimate to the experimental setup. During this period, we encouraged participants to look at the plate and spoon gripper and to open a snack container with their hands. Participants were then given 10 minutes to practice controlling the robot with the joystick and manipulating some of the objects in the experimental environment. We encouraged participants to practice manipulating the snack container since we assumed that it would be the most challenging object to manipulate. The eye tracker was then adjusted for the participant and calibrated before trials.

Prior to beginning the trials, participants were informed of the 3 possible outcomes which would end a trial:

1) The participant met all the criteria for task completion.
2) The plate, spoon gripper, or the container with the required snack were no longer in the robot's workspace (out of reach).
3) The participant wanted to end the trial early.

The instructions given to the participants before each trial consisted of the task's completion criteria:

- Three scoops of the specified snack must be transferred to the plate.
- The plate containing the food must be on the counter.
- A scoop consists of at least one piece of the food.

The only portion of the instructions that changed between trials was the specified snack, a condition which was randomized across trials.

Because of the complexity of the task, mistakes by participants that left objects in challenging positions were common. In cases of these mistake, participants were allowed to attempt to overcome the mistake. In one case, the participant broke the snack container, and the broken container was replaced with a different snack container rather than ending the trial. Additionally, participants often collided the robot into objects in the environment, including the counter and the cabinet. To prevent damage to the robot, a low-volume beep would set off if the torque on the robot's joints surpassed a certain threshold, and the robot would stop taking commands from the joystick if the torque surpassed a higher threshold. In this event, the trial was paused, and the robot was reset to its initial position. If the robot's end effector was gripping an object at the time of collision, the object was placed back into the end effector after reset.

C. Participants

Through initial pilot testing, we found that this task took about 20 minutes to complete, even for users familiar with the robot's operation. Due to the difficulty of the task we designed, we chose to recruit 9 able-bodied participants (3 female, 6 male) who are familiar with robotics in order to mitigate some of the challenges associated with the novice user experience. We wanted our participant pool to be more likely to quickly adapt to teleoperating a robot with a joystick and to overcoming challenging robot configurations. Additionally, because the cabinet and counter make robot collisions highly possible, we wanted our participants to be especially mindful of the possibility of damage to the robot.

IV. PRELIMINARY RESULTS AND DISCUSSION

At this stage in the study, we are still in the process of analyzing the data. While the ultimate goal of this project is to characterize eye gaze behavior during the teleoperation of a robot in a multi-stage task, we can only discuss interesting features we have seen across participants. As such, this section describes some of those interesting features and the future directions of the project.

A. Initial Analysis

Of the 9 participants, we excluded 4 participants' data from further analysis because they experienced difficulty performing the task without glasses. Another participant's data was excluded from analysis because of poor gaze tracker calibration. From the remaining 4 participants, we collected around 1.5 hours of usable data. This data represents 7 trials, 3 of which include task completion. In the other 4 trials, 2 failed due to an object dropping out of the robot's workspace. The other 2 failed because the participant opted to end the trial and because of a software malfunction, respectively. The duration of the failed trials ranged from 3 minutes to 26 minutes, and the successful trials ranged from 12 minutes to 23 minutes.

B. Preliminary Observations

Aronson et al.'s [9] previous work involving eye gaze behavior during the teleoperation of a robot in a single task (spearing a marshmallow with a fork) showed that most fixations during the task were on the robot's end effector. These fixations, coined "monitoring glances" by the authors, were used by participants to keep track of the position of the end effector throughout the task. From a qualitative review of the data, we also found that most fixations were on the robot's end effector during teleoperation in our study's multi-stage task. However, we also found more distinct gaze patterns in our study than in [9]. This outcome was what we hypothesized since the task we designed is significantly more difficult to complete than spearing a marshmallow. This hypothesis was largely influenced by psychology studies into hand-eye coordination which found unique gaze patterns during complex tasks, suggesting that eye gaze provides insight into people's strategies for task completion [12], [14], [18], [27].

One feature that we expected to see across participants is look-ahead fixations, a gaze behavior first defined by Pelz et al. [8] as a fixation on an object several seconds prior to a person reaching for the object. To bring this concept into the context of robot teleoperation, Aronson et al. [9] coined this behavior a "planning glance." During the trials, we found many instances of planning glances during transitions between subtasks. However, we also found many cases of planning glances while participants were in the middle of the subtask. Fig. 5 provides an example of this kind of planning glance. As shown, the participant fixated on the the gripper of the candy container, the Aruco tag above it, and the actual container, all while the participant was attempting to set the spoon gripper down and release it. After releasing the spoon gripper, the participant transitioned to the subtask of retrieving the candy container. While we originally expected planning glances during transitions, we were surprised to see planning glances occur even earlier than the transition period.

Another noteworthy instance of planning glances occurred while the participants transferred food from the snack container to the plate. These glances were often to the object that the participant intended to move the spoon to next (i.e. to the container to obtain a spoonful of food or to the plate to dump the food). All of these planning glances can be powerful sources of data for anticipating the next subtask in an ADL.

Although a primary eye gaze behavior discussed in hand-eye coordination literature about strategy inference is look-ahead fixations, we also expected to see eye gaze behavior specific to teleoperating a robot. A major challenge of teleoperation is maneuvering a robot through difficult joint configurations to achieve a goal, whether that be to reach a certain point in the robot's workspace or to move the robot into an ideal configuration for object manipulation. These difficult joint configurations are often caused by reaching any of the robot's joint limits, its workspace limit, or an internal collision (when one part of the robot collides with

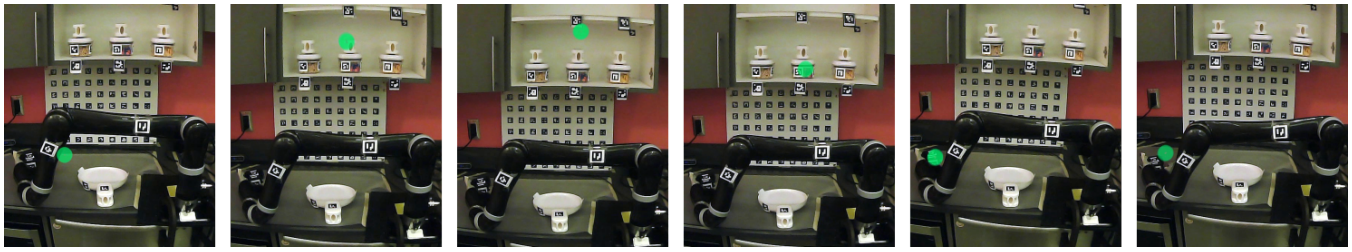


Fig. 5. Planning glance to candy while participant places and releases spoon gripper.

another part). Fig. 6 shows some examples of difficult robot configurations encountered by a participant (during our initial pilot study) while attempting to place the plate on the counter.

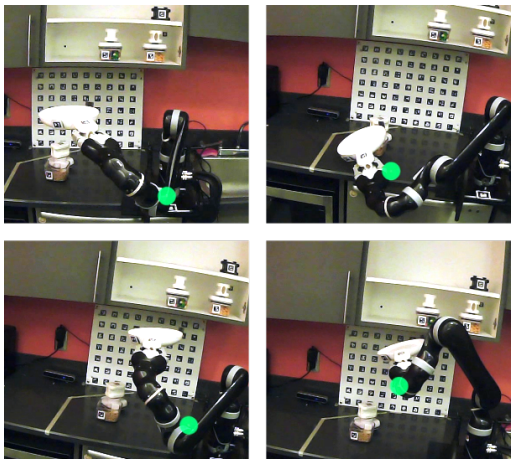


Fig. 6. Challenging robot configurations during plate placement.

Overcoming challenging robot configurations is a key aspect of this study's multi-stage task since the cabinet and counter obstruct portions of the robot's workspace. To accomplish a goal despite difficult configurations, the participants in this study glanced at the robot's joints, a finding also reported in [9]. Fig. 7 shows an example of this behavior as the participant (from our pilot study) attempted to adjust the position of the robot's joints in order to set the plate down on the counter. In the study, we found that many participants faced these configuration issues when placing objects on the counter or retrieving objects on the bottom right or top left of the cabinet. In all of these scenarios, glances to the joints of the robot were present.

Fig. 7. Glances to robot's joints to overcome challenging robot configuration.

Although gaze is the primary focus of this study, the relationship between eye gaze and head movements is an interesting factor in object manipulation studies in psychology [27]. Because the robot visually occluded some objects during manipulation, we saw participants tilt their heads to overcome that occlusion to monitor the position of the end effector in relation to an object. Fig. 8 shows a participant tilt their head to see the gripper on the snack container which was previously occluded from the field of view by the end effector.

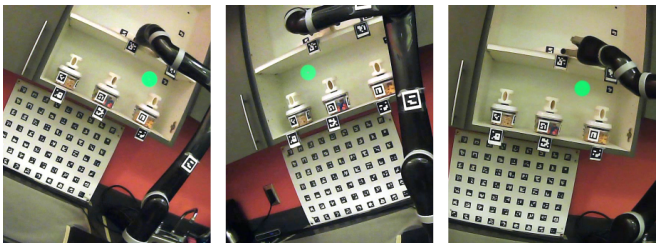


Fig. 8. Head movements to overcome visual occlusion.

In summary, we found that the majority of eye gaze behavior was monitoring glances, in which the participants tracked the position of the end effector, but that planning glances emerged that provided insight into participants' next intended subtask during the trial. We also saw a high frequency of glances to the robot's joints as participants attempted to grasp objects or transfer objects to different locations, particularly between the cabinet and the counter. Finally, participants often tilted their heads when the end effector was about to interact with an object.

C. Future Work

Once we have finished analyzing the data from this study, we intend to use features such as planning glances to build a classifier that can anticipate the next subtask during an ADL. After training and testing the classifier, we intend to test that classifier in a user study to evaluate its implementation in the real world. Then, we want to explore the potential for implementing the classifier into the state-of-the-art shared autonomy framework described in [4].

V. CONCLUSION

In this preliminary work, we began to investigate the potential use of eye gaze for anticipating the progression of an ADL. We described the design of a data collection study in which participants perform a food preparation task with a joystick-controlled robot arm while eye gaze data is

collected. Then, we discussed our preliminary observations from the study. Ultimately, we look forward to identifying strategies to make algorithms in assistive robotics robust to ADLs through the use of natural gaze patterns.

ACKNOWLEDGMENTS

This work was supported by the Robotics Institute Summer Scholars Program and the National Science Foundation. 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.

REFERENCES

[1] Kinova Robotics, "Kinova - Robotic Arm Series." https://www.kinovarobotics.com/en/products/robotic-arm-series, 2018. Retrieved Sept. 6, 2018.

[2] Exact Dynamics, "iARM." http://www.exactdynamics.nl/site/?page=iarm, 2018. Retrieved Sept. 6, 2018.

[3] 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)*, pp. 35–42, March 2016.

[4] S. Javdani, J. A. Bagnell, and S. S. Srinivasa, "Shared autonomy via hindsight optimization," *CoRR*, vol. abs/1503.07619, 2015.

[5] 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, 2018.

[6] H. Admoni and S. Srinivasa, "Predicting User Intent Through Eye Gaze for Shared Autonomy," *AAAI Fall Symposium Series: Shared Autonomy in Research and Practice (AAAI Fall Symposiums)*, pp. 298–303, 2016.

[7] R. S. Johansson, G. Ran Westling, A. Bä, and J. R. Flanagan, "Eye-Hand Coordination in Object Manipulation," *Journal of Neuroscience*, vol. 21, pp. 6917–6932, sep 2001.

[8] J. B. Pelz and R. Canosa, "Oculomotor behavior and perceptual strategies in complex tasks," *Vision Research*, vol. 41, pp. 3587–3596, nov 2001.

[9] 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*, (New York, New York, USA), pp. 4–13, ACM Press, 2018.

[10] 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, pp. 1311–1328, nov 1999.

[11] M. Hayhoe, "Vision Using Routines: A Functional Account of Vision," *Visual Cognition*, vol. 7, pp. 43–64, jan 2000.

[12] 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, p. 6, feb 2003.

[13] M. F. Land and D. N. Lee, "Where we look when we steer," *Nature*, vol. 369, pp. 742–744, jun 1994.

[14] N. Mennie, M. Hayhoe, and B. Sullivan, "Look-ahead fixations: anticipatory eye movements in natural tasks," *Experimental Brain Research*, vol. 179, pp. 427–442, may 2007.

[15] D. H. Ballard, M. M. Hayhoe, and J. B. Pelz, "Memory Representations in Natural Tasks," *Journal of Cognitive Neuroscience*, vol. 7, pp. 66–80, jan 1995.

[16] M. M. Hayhoe, D. G. Bensinger, and D. H. Ballard, "Task constraints in visual working memory," *Vision Research*, vol. 38, pp. 125–137, jan 1998.

[17] M. Hayhoe and D. Ballard, "Eye movements in natural behavior," *Trends in Cognitive Sciences*, vol. 9, pp. 188–194, apr 2005.

[18] M. Hayhoe and D. Ballard, "Modeling task control of eye movements.," *Current biology : CB*, vol. 24, pp. R622–8, jul 2014.

[19] H. Admoni and B. Scassellati, "Social Eye Gaze in Human-Robot Interaction: A Review," *Journal of Human-Robot Interaction*, vol. 6, p. 25, mar 2017.

[20] A. Haji Fathaliyan, X. Wang, and V. J. Santos, "Exploiting three-dimensional gaze tracking for action recognition during bimanual manipulation to enhance humanrobot collaboration," *Frontiers in Robotics and AI*, vol. 5, p. 25, 2018.

[21] P. Schydlo, M. Rakovic, L. Jamone, and J. Santos-Victor, "Anticipation in Human-Robot Cooperation: A Recurrent Neural Network Approach for Multiple Action Sequences Prediction," feb 2018.

[22] 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, pp. 846–851, IEEE.

[23] N. F. Duarte, M. Rakovi, J. Tasevski, M. I. Coco, A. Billard, and J. Santos-Victor, "Action anticipation: Reading the intentions of humans and robots," *IEEE Robotics and Automation Letters*, vol. 3, pp. 4132–4139, Oct 2018.

[24] R. M. Aronson and H. Admoni, "Gaze for Error Detection During Human-Robot Shared Manipulation," *RSS Worksho: Towards a Framework for Joint Action*, 2018.

[25] 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)*, pp. 83–90, IEEE, mar 2016.

[26] Pupil Labs, Inc., "Pupil Labs - Pupil." https://pupil-labs.com/pupil/, 2018. Retrieved Aug. 8, 2018.

[27] E. Kowler, Z. Pizlo, G.-L. Zhu, C. J. Erkelens, R. M. Steinman, and H. Collewijn, "Coordination of Head and Eyes during the Performance of Natural (and Unnatural) Visual Tasks," in *The Head-Neck Sensory Motor System*, pp. 419–426, Oxford University Press, apr 1992.

Efficient Redistribution of Heterogeneous Swarm Robots for Building Security

Emily Duan[1], Sha Yi[2] and Katia Sycara[3]

***Abstract*— Swarm robotics is the coordination of multi-robot systems that was first inspired by natural swarms. Although each individual robot may have limited capabilities, a collective number of these robots can display great flexibility and robustness in groups, which is valuable for complex behaviors. In a community of heterogeneous swarm robots, each type of robot has its own unique capabilities. Given these different capabilities, the need to adapt to specific needs of the desired task and the ability to allocate these robots efficiently based on their strengths is of interest. Current literature only addresses a method to optimize the redistribution of heterogeneous swarm robots based on the presence or absence of these capabilities. However, the ability to quantify swarm robot capabilities can increase the efficiency of the redistribution of robots. To combat this deficit, we are utilizing an existing method to redistribute heterogeneous swarm robots by quantifying the capabilities present in the swarm.**

I. INTRODUCTION

Advancements in embedded systems are enabling the use of large-scale robot systems, i.e. swarm robots, to accomplish laborious tasks. However, as we encounter increasingly complex tasks, homogeneous robot systems are unable to accommodate all facets of the problem and the inadequacy of the capabilities on these large-scale robot systems are revealed. This has led to a shift in focus on the formulation of robot systems with multiple types of robots, for example Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs). [2,3] Current methods of task allocation of heterogeneous swarm robots can identify the presence or absence of capabilities in the swarm, which is a step toward efficiently utilizing these multi-agent systems. However, knowledge of the presence or absence of capabilities may be insufficient to distribute robots efficiently. With more complex behaviors for heterogeneous swarm robots, robot swarms may need to know the quantity of capabilities present to reallocate themselves optimally.

In this work, our objective is to find an optimal policy to distribute a swarm of heterogeneous robots as quickly and efficiently based on the quantity of the capabilities present in the swarm. An example could be if an Unmanned Ground Vehicle (UGV) has three cameras, one Lidar and two infrared while an Unmanned Aerial Vehicle (UAV) only has one camera. Depending on the requirements of the specified task, the allocation of these robots will differ based on the desired needs of the task. This problem formulation was inspired by the tactic for establishing a perimeter around a building for the DARPA Sprint project. In the case with monitoring a building, the doors and windows are identified as locations of higher interest, where the capabilities of Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs) would be able to monitor these regions respectively. However,the perimeter formulation around the entire building may be unnecessary if certain sides of the building do not have doors or windows. Furthermore, there may be sides of the building where there are multiple doors and windows that need to be monitored.

Our problem is a case of the *MT-MR-TA: Multi-Task Robots, Multi-Robot Tasks* problem [4], which is described as a strong NP hard task allocation problem [5] that divides the robots in a multi-robot system into subsets of robots and pairs the subset of robots to tasks. Previous algorithms have been shown to be unsuitable for large number of robots. The work presented in [1] takes the scalability and capabilities of the robots into consideration for the robots to adapt to changes in the swarm. The current work focuses on optimal transition rates for the heterogeneous swarm robots to quickly reach the desired trait distribution configuration. [6,7,8]

The work presented in [1] focuses on the optimization of transition rates that permit a heterogeneous robot swarm to converge quickly to a configuration that satisfies a desired trait distribution. The improvement from the previous work[2,6,8] is that we modified the species matrix to depict the quantity of the capabilities present in the swarm instead of solely identifying if the capability is present or absent to effectively distribute the robots. Contrary to previous work [4, 6] on optimization methods for this formulation, the final robot distribution is not used or in other words, the user will simply specify how much of a given capability is needed for a given task. We will utilize the optimization methods for transition rates and convergence times shown in [1] to reallocate the robots based on the quantity of capabilities present in the swarm.

II. RELATED WORKS

Our problem formulation was heavily based on the work presented in [1] with the exception to the binary instantiations of traits that correspond to the presence or absence of given traits in the species. The authors of [1] modeled the swarm of heterogeneous robots as a community, where each species in this community is a different type of robot. A unique set of traits are defined for each species of robots.

[1]E. Duan is a Mechanical Engineering student in the College of Engineering and Information Technology at the University of Maryland, Baltimore County.

[2]S. Yi is a Masters of Science in Robotics student in the Robotics Institute at the Carnegie Mellon University.

[3]K. Sycara is a professor in the Robotics Institute, School of Computer Science at Carnegie Mellon University.

A. Notation

We have utilized the same notation as presented in [1] to set up the problem. The community of S robot species contains a total number of robots N, where there are $N^{(s)}$ robots per species and $\sum_{s=1}^{S} N^{(s)} = N$. A set of U traits is defined for the community and each robot species owns a subset of these traits. A species is defined by a vector $q^{(s)} = [\, q_1^{(s)}, q_2^{(s)}, ..., q_U^{(s)}]$. The matrix $\mathbf{Q}$ is defined by $S \times U$, where $\mathbf{q}^{(s)}$ are the rows. The interconnection topology of the M tasks was modeled by a directed graph, $G = (\mathcal{E}, V)$, where V, represents tasks $\{1,...,M\}$ and the set of edges, $\mathcal{E}$, represents the ordered pairs (i,j) where $(i,j) \in V \times V$. The edges indicate the possibility to switch between the two adjacent tasks. The graph G is assumed to be a strongly connected graph and the robots have knowledge of this graph. Every edge $\mathcal{E}$ is assigned a transition rate, $k_{i,j}^{(s)}$, which defines the transition probability per unit time for one robot of species s at task i to switch to task j. This transition rate is defined where $k_{ij}^{(s)} > 0$ and a limitation is imposed on the maximum rate of each edge with $k_{ij}^{(s)} < k_{ij,\,max}^{(s)}$. System identification methods on the actual setup can be applied to the determination of these values as well as observable factors. The distribution of the robots in a species s at time t is depicted as a vector $x^{(s)} = [\, x_1^{(s)}(t), ..., x_M^{(s)}(t)]^T]$, where $x^{(s)}$ are the columns of $\mathbf{X(t)}$ matrix. The distribution of traits among the tasks $\mathbf{Y}$ is depicted by the $M \times U$ matrix. This relationship with respect to time t is given by

$$\mathbf{Y(t)} = \mathbf{X(t)} \cdot \mathbf{Q} \tag{1}$$

B. Problem Statement

Initial state of the robot system and the initial distribution of traits among the tasks is described by $\mathbf{X(0)}$ and $\mathbf{Y(0)}$ respectively.

$$\frac{dx_t^{(s)}}{dt} = \sum_{\forall j|(i,j)\in\mathcal{E}} k_{ji}\mathbf{x_t^{(s)}}(t) - \sum_{\forall j|(i,j)\in\mathcal{E}} k_{ij}\mathbf{x_t^{(s)}(t)} \tag{2}$$

The base model for all species is given by

$$\frac{dx^{(s)}}{dt} = \mathbf{K^{(s)}x^{(s)}} \qquad \forall_s \in 1, ..., S \tag{3}$$

where $\mathbf{K}^{(s)} \in \mathsf{R}^{M \times M}$ is a rate matrix with the following properties

$$\mathbf{K}^{(s)T}1 = 0 \tag{4}$$

$$\mathbf{K}_{ij}^{(s)} \geq 0 \qquad \forall (i,j) \in \mathcal{E} \tag{5}$$

The result of these two properties is the following definition:

$$\mathbf{K}_{ij}^{(s)} = \begin{cases} k_{ji}^{(s)}, & \text{if } i \neq j, (i,j) \in \mathcal{E} \\ 0, & \text{if } i \neq j, (i,j) \ni \mathcal{E} \\ -\sum_{t=1,(i,j)\in\mathcal{E}}^{M} k_{ij}^{(s)} & \text{if} i = j \end{cases}$$

The total number of robots and the number of robots per species is conserved and there Eq. 3 is subject to the following constraints

$$\mathbf{X}^T \cdot 1 = [N^{(1)}, N^{(2)}, ..., N^{(S)}]^T \tag{6}$$

$$with \quad \mathbf{X} \succeq 0, \tag{7}$$

where $\succeq$ is an element-wise greater-than-or-equal-to operator. Given a target distribution $\bar{\mathbf{Y}}$, the goal is to find an optimal rate matrix $\mathbf{K}^{(s)*}$ for each species s so that

$$\bar{\mathbf{Y}} = \bar{\mathbf{X}} \cdot \mathbf{Q} \tag{8}$$

The objective is to redistribute the robots of each species configured based to the initial distribution of the robots, $\mathbf{X}(0)$ to reach a desired trait configuration $\bar{\mathbf{Y}}$. In this process, a robot configuration $\bar{\mathbf{X}}$ that satisfies Eq. 1 will be reached. It is noted that there may be several $\bar{\mathbf{X}}$.

III. Methodology

Our methodology follows the methodology discussed in detail in [1] for acquiring an optimal transition matrix $\mathbf{K}^{(s)*}$ for each species to reach a desired trait distribution. The binary instantiations of traits that correspond to the presence or absence of given traits in the species is replaced with a quantity of the number of capabilities or traits that are present in the species. The species matrix $\mathbf{Q}$ is given as

$$\mathbf{Q} = \begin{bmatrix} u_{1,1} & \cdots & u_{1,U} \\ \vdots & \ddots & \vdots \\ u_{s,1} & \cdots & u_{s,U} \end{bmatrix} \tag{9}$$

where u is the quantity of a certain capability in the species, s is the number of species in the swarm and U is the number of capabilities present in the species. As presented in [1], a differentiable objective function can be efficiently minimized through gradient descent techniques and this method explicitly minimizes the convergence time of $\mathbf{K}^{(s)}$.

A. Optimal Transition Rates

To obtain the values of $\mathbf{K}^{(s)*}$, the error, $\mathbf{E}$, is considered and given by

$$\mathbf{E} = \bar{Y} - \mathbf{Y}(t) \tag{10}$$

where $\mathbf{Y}(t)$ is the solution of the linear ordinary differential equations, Eq. 3 and Eq. 8, and given by

$$\mathbf{Y}(t) = \sum_{s=1}^{S} e^{\mathbf{K}^{(s)*}\tau}\mathbf{x}_0^{(s)} \cdot \mathbf{q}^{(s)} \tag{11}$$

where τ is defined as the time when the desired distribution is reached. The optimization transition rates problem is formulated as

$$minimize \quad \mathcal{J}^{(1)} = ||\mathbf{E}||_F^2 \tag{12}$$

such that $k_{ij}^{(s)} < k_{ij,max}^{(s)}$. A minimum cost is found when the final trait distribution coincides with the desired trait distribution that contingent to the maximum transition rates $k_{ij,\,max}^{(s)}$. The operator $\|\cdot\|_F$ is the Frobenius norm of a matrix. The gradient descent of the cost function is given as

$$\frac{\partial \mathcal{J}^1}{\partial \mathbf{K}^{(s)}} = V^{-1^T}\left[V^T \frac{\partial \mathcal{J}^{(1)}}{\partial e^{\mathbf{K}^{(s)}\tau}} \odot \mathbf{W}(\tau)\right]V^T\tau \tag{13}$$

where $\odot$ is the Hadamard product, $\mathbf{K}^{(s)}$ = VDV^{-1} is the eigendecomposition of $\mathbf{K}^{(s)}$. V is the $M \times M$ matrix with the jth column as a right eigenvector corresponding to eigenvalue d and D = diag(d_1, ..., d_M). The derivative of the cost function with respect to the expression $e^{\mathbf{K}^{(s)}\tau}$ is given as

$$\frac{\partial \mathcal{J}^{(1)}}{\partial e^{\mathbf{K}^{(s)}}\tau} = -2\mathbf{E} \cdot \left[\mathbf{x}_0{}^{(s)} \cdot \mathbf{q}^{(s)}\right]^{\mathrm{T}} \tag{14}$$

Further details on the gradient descent of the cost function can be found in [1].

B. Optimal Convergence Time

The optimal convergence time problem is formulated as follows

$$\mathbf{K}^{(s)*}, \mathbf{t}^{*} = \underset{\mathbf{K}^{(s)},\tau}{\operatorname{argmin}} \mathcal{J}^{(3)}, \tag{15}$$

The cost function from Eq. 9 is modified to consider the convergence time τ as a variable and is given by

$$\text{minimize} \quad \mathcal{J}^{(2)} = \mathcal{J}^{(1)} + \alpha\tau^2 \tag{16}$$

such that $k_{ij}{}^{(s)} < k_{ij,\,max}{}^{(s)}$, $\tau > 0$ and $\alpha > 0$. The importance of th convergence time can be increased by increasing α. The optimization of Eq. 12 will result in the transition rates that would lead to the desired trait distribution quickly but the steady state of $\mathbf{K}^{(s)}$ is not guaranteed. If we strive to compute the transition rates and ensure the constant state reached at the optimal time **t***, the cost function in Eq. 12 is modified as follows.

$$\min \quad \mathcal{J}^{3} = \mathcal{J}^{2} + \beta \sum_{s=1}^{S} \parallel e^{\mathbf{K}^{(s)}(\tau)}\mathbf{x}_0{}^{(s)} - e^{\mathbf{K}^{(s)}(\tau+\nu)}\mathbf{x}_0{}^{(s)} \parallel_2{}^2 \tag{17}$$

such that $k_{ij}{}^{(s)} < k_{ij,\,max}{}^{(s)}$, $\tau > 0$ and $\beta > 0$. The additional term included in the cost function permits the state reached after applying $\mathbf{K}^{(s)*}$ remains steady for an arbitrarily long time interval ν. As the value of β increases, the difference in robot distributions at τ and $\tau + \nu$ decreases, which indicates that as β increases, the trait distribution corresponding to the steady state of $\mathbf{K}^{(s)}$ reaches close to the desired trait distribution $\bar{Y}$. Further details on the gradient descent of the cost function can be found in [1].

C. Implementation

For a community of 10 heterogeneous swarm robots (N = 10), there are two species (S = 2) ($N^{(s)}$ = 5), 2 traits (U = 2) and 5 tasks (M = 5) present. The initial distribution of robots, $\mathbf{X}_0$, is set as

$$\mathbf{X}_0 = \begin{bmatrix} 5 & 5 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{18}$$

and the desired trait distribution, $\bar{\mathbf{Y}}$, is as follows

$$\bar{\mathbf{Y}} = \begin{bmatrix} 1 & 2 \\ 1 & 0 \\ 1 & 2 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{19}$$

Q equations We are currently trying to implement the code in Python using NumPy and SciPy libraries.

IV. Results

Once we obtain results from our implementation, we hope to compare our results and evaluate our performance with the results presented in [1]. We will update this paper with the results after the implementation is completed.

V. Conclusion

We present an optimal policy for redistributing heterogeneous swarm robots among a set of tasks based on the quantity of capabilities present in the swarm to reach the desired distribution of capabilities among the tasks. We utilize the optimization problem and solution based on an analytical gradient presented in [1] to efficiently compute the transition rates and convergence times for distribution. We believe that this method will be well-suited for applications that require additional knowledge of the capabilities present in the control and coordination of large-scale team of robots to quickly reach the desired configuration. This objective is only a piece of a larger vision to develop control and coordination strategies for teams of heterogeneous robots with specific capabilities given that the capabilities of each individual robot may vary in performance. Variations of this problem can also be applied to disaster relief missions or for building security.

Acknowledgements

Emily would like to thank the Robotics Institute REU for funding this work. She would especially like to thank Dr. Katia Sycara and Sha Yi for their guidance in this project. She would also like to thank Rachel Burcin, John Dolan and the Robotics Institute Summer Scholars team for their support.

References

[1] A. Prorok, M. A. Hsieh, V. Kumar, Fast Redistribution of a Swarm of Heterogeneous Robots, International Conference on Bio-inspired Information and Communications Technologies, 2015

[2] T. Balch and L. E. Parker. Special issue on Hetero- geneous Multi-Robot Systems. Autonomous Robots, 8:207383, 2000.

[3] E. G. Jones, B. Browning, M. B. Dias, B. Argall, and M. Veloso. Dynamically Formed Heterogeneous Robot Teams Performing Tightly Coordinated Tasks. International Conference on Robotics and Automation (ICRA), pages 570575, 2006.

[4] B. P. Gerkey and M. J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. Interntional Journal of Robotics Research, 23(9):939 954, 2004.

[5] B. Korte and J. Vygen. Combinatorial Optimization: Theory and Algorithms. Springer-Verlag, Berlin., 2000.

[6] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar. Optimized Stochastic Policies for Task Allocation in Swarms of Robots. IEEE Transactions on Robotics, 25:927937, 2009.

[7] A. Halasz, M. A. Hsieh, S. Berman, and V. Kumar. Dynamic Redistribution of a Swarm of Robots Among Multiple Sites. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007.
[8] M. A. Hsieh, A. Halasz, S. Berman, and V. Kumar. Biologically inspired redistribution of a swarm of robots among multiple sites. Swarm Intelligence, 2(2-4):121 141, 2008.

Planning with Unreliable Controllers

William Edwards
Department of Computer Science
University of South Carolina
wre@email.sc.edu

Dhruv Saxena
Robotics Institute
Carnegie Mellon University
dhruvsaxena@cmu.edu

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
maxim@cs.cmu.edu

***Abstract*— State lattice-based planning has been used for motion planning in a wide variety of robots, including ground, water, and air vehicles. The State Lattice with Controllers (SLC) framework expands upon traditional methods by allowing planning with controllers, such as wall-following or visual servoing, in addition to metric motion primitives. This allows SLC to plan in environments where execution of metric motion primitives is unreliable, such as GPS-denied areas. However, the SLC framework assumes that each controller reliably navigates the robot to a single end state. In practice, it may be useful to reason about controllers which are unreliable (*i.e.* the robot may end up in one of many states after executing the controller). This work proposes an algorithm for planning with arbitrarily unreliable controllers. Preliminary experimental results evaluate the performance of the algorithm in simulation.**

I. Introduction

Traditional search-based planning methods for robotics depend on the ability of the robot to execute metric motion primitives. Unfortunately, there are many situations in which metric motion primitives cannot be executed reliably. For example, a robot operating in a GPS-denied environment might not be able to perform state estimation with the accuracy needed for reliable execution of metric motion primitives. One method proposed to solve this issue is the State Lattice with Controllers (SLC) [1]. The SLC framework attempts to solve this problem by permitting the planner to choose between metric motion primitives and controllers such as wall-following or visual servoing. These controllers may be able to operate in regions where metric motion primitives cannot be executed reliably.

One limitation of SLC is that it requires controllers which can reliably guide the robot to a particular state. However, it is sometimes desirable to execute an unreliable controller, that is, a controller which may guide the robot to one of many states from the same starting state. In some situations, executing an unreliable controller could allow for a more direct path.

A. Motivating Example

Consider the scenario depicted in Figure 1. A flying robot must navigate from the start to the goal, but there is a variable wind blowing across the middle of the room. The robot could reach the goal region by executing a series of wall-following controllers around the boundary of the environment, but this is not the shortest path. Alternatively, the robot could execute a simple proximity controller, flying in the same direction until it senses an object ahead. If the robot executes a proximity controller in the north direction, it could end up in many states since the wind is variable and the proximity controller does not use feedback to stabilize the trajectory. However, so long as the robot ends up somewhere along the top wall, it can then execute a wall-following controller to reach the goal region. Executing the unreliable proximity controller followed by the reliable wall-following controller gives a path which is shorter than executing a series of reliable wall-following controllers.

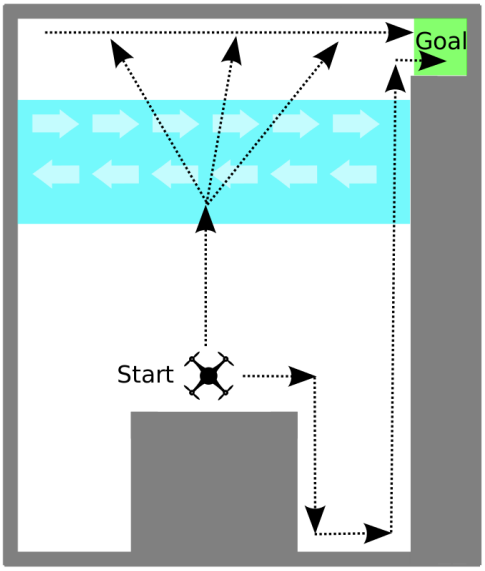


Fig. 1: Traveling from the start to the goal can be done either by following the wall around the boundary or by flying across the room until a wall is detected, and then following the wall into the goal region. In the latter case, the variable wind (blue) will cause the robot to end up in different positions along the top wall, but the robot is still able to reach the goal. Our method permits this temporary uncertainty.

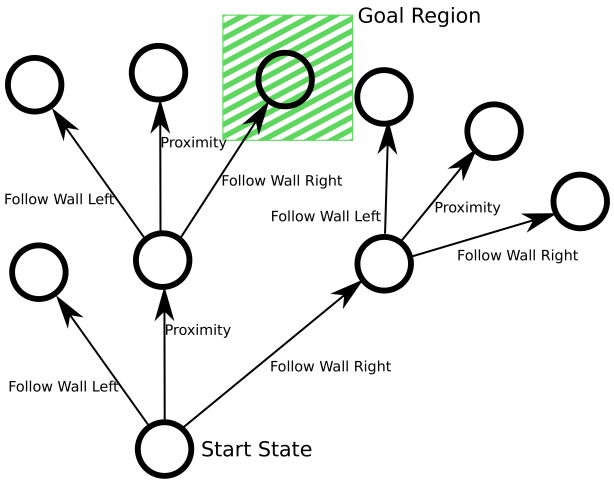


Fig. 2: The space of controller/trigger sequences is represented by a tree, with each edge corresponding to a controller/trigger pair, and each node corresponding to a sequence of controllers and triggers applied in order.

B. Overview

In this work, we present a novel algorithm which allows for planning with unreliable controllers and provides a probabilistic guarantee on the optimality and correctness of the solution. In Section II, we provide an overview of the relevant literature. In Section III, we formally define the problem, and in Section IV, we provide a detailed description of the algorithm. Experimental results are given in Section V and discussion is provided in Section VI.

II. Related Work

There is a large, existing body of work on planning under motion and sensing uncertainties. The *LQR-Trees* algorithm [2] approaches the problem of motion uncertainty by constructing a series of feedback controllers which guide the robot toward a feasible trajectory from any state in the region. One limitation of this approach is that it requires an estimate of the full state in order to provide controller feedback. The *LQG-MP* (Linear Quadratic Gaussian - Motion Planning) [3] is an algorithm which can explicitly account for both motion and sensing uncertainties, but is limited to handling Gaussian noise.

There is also some existing literature on planning for unreliable robots with very limited sensing capabilities. [4] solves the problem of active and passive localization with a robot which only has a clock and a contact sensor. [5] extends the work to create provably correct plans for a blind robot with bounded uncertainty. However, both of these methods are limited to a specific model.

III. Problem

A. Definitions

Let S denote the set of possible states a robot can take, and let $\mathscr{C}$ denote the set of available controllers. Each controller terminates either upon its natural completion (*e.g.* a wall-following controller reaching the end of a wall) or upon the detection of some perceptual trigger (*e.g.* a wall-following controller detects a window or opening in the wall). We use $\mathscr{T}$ to denote the total set of perceptual triggers and define the mapping $T(c) : \mathscr{C} \rightarrow P(\mathscr{T})$ which gives the set of triggers available for controller c.

Applying a controller c and trigger t to some state $s \in S$ results in a new state and associated cost, which may vary based on noise in the environment, and which is given by the stochastic function $\phi(s, c, t) : S \times \mathscr{C} \times \mathscr{T} \rightarrow S \times \mathbb{R}$. Furthermore, let $\phi'(s, (c_1, \ldots, c_n), (t_1, \ldots, t_n)) \rightarrow S \times \mathbb{R}$ represent the extended stochastic transition function corresponding to the application of $(c_1, \ldots, c_n)$ and $(t_1, \ldots, t_n)$ in order. Given a goal region $G \subset S$, define δ such that

$$\delta(s, (c_1, \ldots, c_n), (t_1, \ldots, t_n)) = \begin{cases} \text{Cost of } \phi'(s, (c_1, \ldots, c_n), (t_1, \ldots, t_n)) & \text{if in } G \\ \infty & \text{o.w.} \end{cases}$$

B. Problem Statement

The problem is, given S, a start state $s \in S$, a goal region $G \subset S$, as well as $\mathscr{C}$, $\mathscr{T}$, T, ϕ, and p, to find a sequence $(c_1, \ldots, c_n)$ and $(t_1, \ldots, t_n)$ which minimizes the p_{th} quantile of $\delta(s, (c_1, \ldots, c_n), (t_1, \ldots, t_n))$. Minimizing the p_{th} quantile, rather than another statistic such as the mean, allows us to account for possibly infinite values of δ and ensures that the probability that the plan guides the robot to the goal region G is at least p.

IV. Algorithm

Algorithm 1 Node Expansion

```
1:  procedure EXPANDNODE(n)
2:      succs ← ∅
3:      for c ∈ 𝒞 do
4:          for t ∈ T(c) do
5:              succ ← node which applies (c, t) to n
6:              Compute LCB_best(succ)
7:              succ ← succs ∪ succ
8:          end for
9:      end for
10:     return succs
11: end procedure
```

In addition to the planning problem, the algorithm accepts ϵ, and α, and utilizes an admissible heuristic h. It guarantees that the solution returned is ϵ-optimal at least $1-\alpha$ of the time. At a high level, our approach is a

Algorithm 2 Main Loop

```
1: procedure MAIN
2:     Open ← {n_s}                                  ▷ n_s is start node
3:     current_solution ← Fast solution with inflated heuristic
4:     LCB_best(n_s) ← h(s)
5:     while not HasCompleted() do
6:         n ← node with minimum LCB_best from Open
7:         Test if p_th quantile of n is finite
8:         if finite then
9:             Compute UCB_δ(n)
10:            Do comparison test to current_solution
11:            if n is better than current_solution then
12:                current_solution ← n
13:            end if
14:        else
15:            succs ← ExpandNode(n)
16:            Add succs to Open
17:        end if
18:    end while
19:    return ReconstructPath(current_solution)
20: end procedure
```

Algorithm 3 Termination Condition

```
1: procedure HASCOMPLETED
2:     if ∀n ∈ Open LCB_best(n) ≤ (1/ε) UCB_δ(current_solution)
   then
3:         return true
4:     else
5:         return false
6:     end if
7: end procedure
```

branch-and-bound algorithm. Possible controller/trigger sequences are represented as a tree (Figure 2) with each edge representing a controller/trigger pair and each node representing a series of controller/trigger applications.

The algorithm proceeds by expanding the leaf nodes of the tree (Algorithm 1). At each expansion, a successor is selected for each available controller/trigger pair. Then, for each successor, the full sequence of controllers is simulated multiple times to obtain a sample of trajectories. For each of these trajectories, a cost and heuristic value is calculated, which is then used to establish a lower confidence bound on the best possible sequence which contains that successor (denoted LCB_{best}). The exact number of simulations per node depends on the confidence specified by the user.

In the main function (Algorithm 2), an initial, sub-optimal solution is first found by searching with an inflated heuristic. An upper bound on the cost of this solution (denoted UCB_δ) is established by running multiple simulations. Next, leaf nodes are expanded in order of increasing LCB_{best}. This process continues until the minimum LCB_{best} value of a leaf node is at least $\frac{1}{\epsilon}$ times UCB_δ for the current best solution (Algorithm 3). This guarantees (with respect to the given confidence level) that the solution returned is no more than ϵ times the optimal solution.

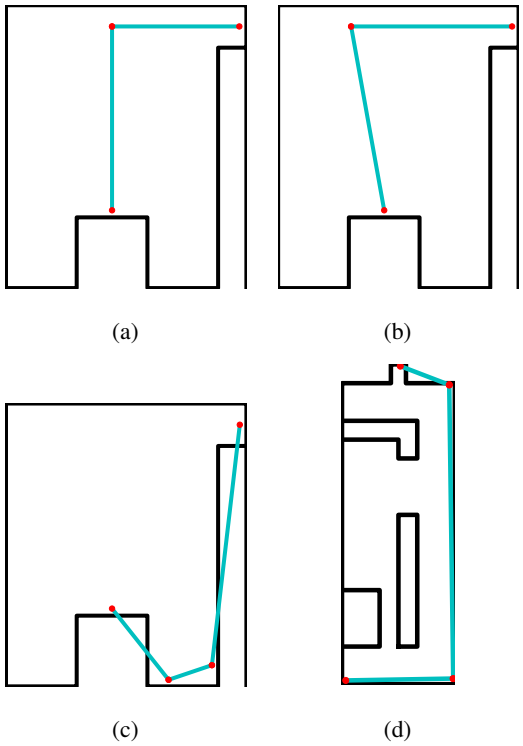


Fig. 3: *(a)* In the small environment (17 x 20 m), the optimal solution without noise is to execute a proximity controller followed by a wall-following controller. *(b)* With a low amount of noise (wind), the optimal solution is unchanged. *(c)* With high noise, the optimal solution is to execute a series of wall-following controllers around the boundary. *(d)* For the large environment (30 x 85 m), the optimal solution is a series of wall-following controllers. Note that for all examples, only the endpoints (red) of each controller are drawn, so the path may appear to cut corners.

V. Results

The algorithm was evaluated on the 17 x 20 meter environment depicted in Figure 1 with no wind, low wind, and high wind. For these tests, we used $p = 0.9$, $\alpha = 0.2$, and $\epsilon = 1$ (optimal solution), as well as a reverse Dijkstra's search as the heuristic. With no wind, the optimal solution was to execute the proximity controller followed by the wall-following controller (Figure 3a). For a low amount of wind, the optimal solution was found to be the same (Figure 3b). However, for a high amount of wind, the planner determined that executing the proximity controller would be too unreliable, instead returning a sequence of wall-following controllers which guide the robot around the boundary of the environment (Figure 3c). In each of these tests, the planning time was less than 10 seconds (Table I).

The algorithm was also tested on a larger environment (30 x 85 meters) without noise. An optimal solution

Environment	Planning Time (s)	Cost (90^{th} percentile)	
		LCB	UCB
Small (No Noise)	2.47	22.43	22.43
Small (Low Noise)	2.55	25.28	26.28
Small (High Noise)	7.15	33.15	33.52
Large	609.68	125.61	125.61

TABLE I: The planning times and solution costs for four environments are given. Since the cost can only be approximated, a lower and upper confidence bound are given.

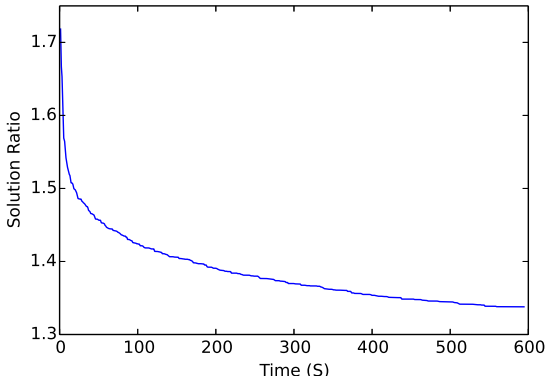


Fig. 4: The ratio, over the course of planning, between the lower bound on the cost of the optimal solution and the upper bound on the cost of the current solution. This ratio represents a bound on the sub-optimality of the best known solution. These results are for the large environment.

was found (Figure 3d), but at approximately 10 minutes (Table I), the planning time required was much larger than for the smaller examples. Figure 4 shows the ratio between the minimum LCB_{best} and the UCB_{δ} of the best known solution over time. This ratio represents a bound on the sub-optimality of the solution as the algorithm progresses.

VI. Discussion

The algorithm as it stands is a work-in-progress and the experimental results presented here are very limited. It is promising that the algorithm is able to find optimal solutions, although the planning time which is currently required for large environments may be prohibitive in some applications. The authors are exploring ideas to improve the algorithm's performance. More tests on a variety of environments under different conditions are also planned.

VII. Acknowledgments

The authors would like to thank the National Science Foundation for their generous financial support. The authors would also like to acknowledge the support of the Robotics Institute Summer Scholars program and its organizers: Dr. John Dolan and Ms. Rachel Burcin.

References

[1] 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," *IEEE International Conference on Intelligent Robots and Systems*, pp. 258–265, 2014.

[2] R. Tedrake, "Lqr-trees: Feedback motion planning on sparse randomized trees," 2009.

[3] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[4] L. H. Erickson, J. Knuth, J. M. O'Kane, and S. M. LaValle, "Probabilistic localization with a blind robot," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1821–1827.

[5] J. S. Lewis and J. M. OKane, "Planning for provably reliable navigation using an unreliable, nearly sensorless robot," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1342–1357, 2013.

Feedback Acquisition From Falling Cat Robot And Its Evaluaction

Kyuto Furutachi[1] and Steven Crews[2]

***Abstract*— Inertial reorientation is an interesting problem, particularly for the application of robots which require midair changes in attitude. Cats reorientation phenomena, which is known as the cat-righting reflex, is said it is one of the best biological models in this field. While theoretical analysis of this phenomenon has been done by many physicists and researchers, few researchers have attempt to replicate this process in a physical robot. We assume that cat's start falling from at rest, then calculate ideal theoretical trajectory following simplified dynamics. Our current robot is following this motion on an open-loop trajectory, i.e. without getting any orientation feedback. However, the state of beginning of its fall might vary on some situations, and some external influence may happen while it's falling. Therefore, the actual trajectory is subject to change due to these influences. In order to enable the robot to correct for differences in initial pose and disturbances, we need to get feedback of its state during free fall.**

I. INTRODUCTION

Cat is able to reorientate its state mid-air when it is falling. This biological phenomena is called as cat-righting reflex. Our basic purpose of this project is replicating this phonomena by creating phisical robot. We apply only two degree actuators as inputs on current robot we have although real cat has more degrees when considering cat-righting reflex. On previous project, we already created phisical robot and calculate ideal trajectory by following dynamix around cat-righting reflex. We assume that a cat is lereased from rest and holizontally. What our current model does is only following calculated trajectory, paticulary stating following angle of actuator which we already calculated earlier. However, cat is not definitely follow our calculated trajectory since something happend in real-world when falling. Therefore, a cat has to be able to modify its trajecotry while it is falling mid-air. Since a cat is falling in less than second which is really fast, the robot has to take an information enough fast to recognize its state every cycle and calculate next behavior which the robot has to follow next cycle.

II. MATERIALS

In this section, we mention the equipment we use for this project and the reason why we chose those staffs. As stated earlier section, we had "planned" trajectory which is already calculated followed theoretical analysis. However, we try to get a feedback from our robot while it's falling every cycle and correct their trajectory so it's able to follow the planned trajectory as close as possible if it's going away from it. In order to do that, the on-board computer should have ability to high-rate calculation for getting feedback with high sample rate and calculate new motion. Also, the actuator should have a big enough torque speed for moving quickly while it's falling. In addition, the IMU sensor should be able to get its orientation while it's falling. Some of the staff we using as an previous model is not good enough for new model. Therefore, we upgrade them so the robot's specification is well above its requirement.

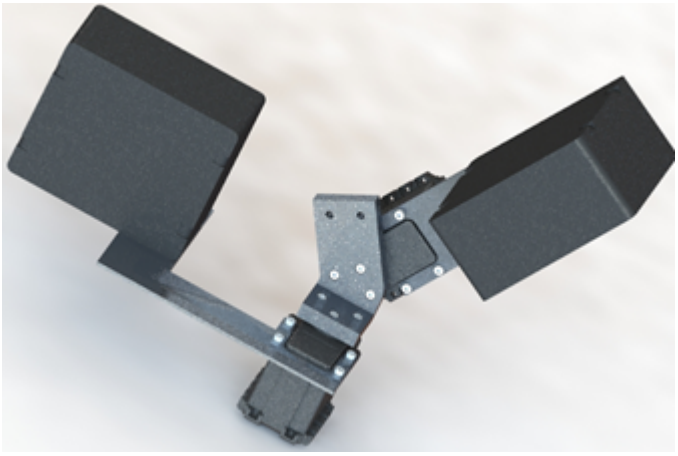


Fig. 1. cat robot model

A. Hardware

As mentioned in introduction, we consider 2 DOF robot for mimicking the cat-righting reflex phenomenon. For actuators, we use Dynamixel servo motor, AX-18A. This servo has nice torque speed which is well above our requirement to move quickly for correcting their behavior. The robot has two black box indicating front and back body of cat, and each box is for a battery or a designed circuit board. These two black boxes are fixed by screw and there are two actuators at the center part of the robot. The figure of our hardware design is shown as Fig. 1 below.

B. Electronics

In the earlier version of this robot, the electric parts we used has several limitations. Therefore we upgrade some of them for getting better performance. First, we mention about limitation of earlier model. Next, we state how we upgrade these pieces.

1) Current limitation:

[1]H. Kyuto Furutatchi is with Faculty of Marine System Engineering. Kyushu University, 744 Motooka Nishi-ku, Fukuoka, Japan 1TE15532W@s.kyushu-u.ac.jp

[2]Steve Crews is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15289, USA screws@andrew.cmu.edu

Computer: Also, we used an Arduino Nano for on-board computer because of its reasonable cost and usability. However, the Nano's 16 MHz clock speed is not fast enough for calculating the next motion referred by feedback.

IMU sensor: In the earlier version of this robot, we used the MPU6050 as a 6-DOF IMU sensor the sample rate of which was 200-300 KHz when we apply DMP library to get quaternions calculated on sensor. DMP stands for "Digital Motion Processor." This is internal calculation library built in MPU 6050 which indicates that MPU6050 becomes to be able to calculate an orientation using sensed information.

2) Improvement:

Computer: As mentioned earlier subsection, Arduino Nano has slow clock rate for our cat robot since cat is falling in quite small amount of time. Therefore, we changed the micro-controller to the much faster and more capable Teensy 3.6. Teensy 3.6 has 180 MHz as a clock speed, which is fast enough for updating the trajectory.

IMU sensor: Also, we get new IMU sensor named as Three Space Sensor Embedded from Yost Lab which is much better than MPU6050 in terms with sample speed, accuracy, facility, and so on. There are couple of built-in filter inside sensor so we have a choice to activate one of the filter or deactivate any filters to get raw values from sensor.

III. METHODOLOGY

A. Getting current state

We need to get 2 kind of state of itself from our falling cat robot. First, we get the sensed orientation of the electronics box while it is falling. Second, the Dynamixels give us the rotation angles of the center joints Since physical relationship among control board, actuators and two black boxes are constant, we can calculate overall system behavior easily.

We estimate the orientation from accelerometer and gyro data with a complementary filter.

The complementary filter is a common filter for estimating an object's orientation using in robotics, aerospace engineering, and similar applications. Simply stating about the definition of complementary filter is that "The complementary filter consists of both low and high pass filter and as it is easier to implement this filter was implemented for getting precise data.[1]" To avoid the prime limitations of Euler Angles (i.e. Gimbal Lock), we use quaternions as the basis for representing orientation. Since we need to predict the next state along the trajectory correctly, we should be able to get an accurate and timely feedback from each IMU sensor and actuators. This will allow us to characterize the real-world behavior in real time. In order to validate our sensed orientation, we used motion capturing system called Optitrack. Optitrack, the brand's software application, provides us with position and orientation information. Data from Optitrack can be considered as ground-based data

and our estimation of orientation get from IMU sensor should be close to it as much as possible. To make sure that our estimation is close enough to ground-based orientation, we use quaternions and compare in terms of it.

IV. RESULTS

On this section, we mainly about how better data the robot could get compared to previous model in terms of couple of points.

A. Sensed IMU v.s. Optitrack

We moved circuite board and made IMU sensor getting quaternion while it is moving. Moreover, we used Optitrack at the same time and getting Optitrack quaternion while it is moving. Fig. 2 shows comparison of quaternion between from IMU sensor and from Optitrack. According to this figure, these two figures are almost same while moving, which indicates that our sensed information can be trusted enough since it is closed to Optitrack's quaternion.

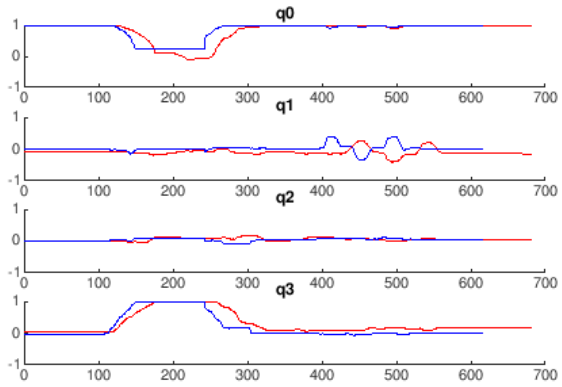


Fig. 2. Quaternion of Sensed v.s. Optitrack

B. Sample rate

Since the time length while our robot is falling is quick which is at least less than 1 seconds, we have to recognize its current state quite fast as well. In order to do that, we really have to care about sample rate which means here is how fast IMU sensor can get next data after getting current data. Also, we should consider so much about how fast micro controller can communicate with actuators and get its current state from them. As Table 1 indicates, we improve the sample rate for getting current state from each of two very much, which seems to be fast enough for our prediction of next behavior while it's falling.

TABLE I
SAMPLE RATE IN MICRO SECONDS

Process	Previous	New
Get quaternion	<7000	<1200
Get raw IMU data	<5000	<1000

V. CONCLUSION

We could make cat robot got an accurate sensed information while it is falling. Also, the sample rate of both IMU information and angle of rotation of actuator is quite fast compared to previous model. Therefore, we are now ready

to tackle applying mathematical algorithm into our model. Then, we will be able to get feedback from cat while it is falling.

ACKNOWLEDGMENT

We thank Dr. Howie Choset, Dr. John Dolan, Ms. Rachel Burcin, the others who support Robotics Institute Summer Scholars for providing Kyuto Furutachi the opportunities to work at Carnegie Mellon University. We also appreciate Japanese government for financial support. We are immensely grateful to Dr. Juliann Reineke and Alex H for their comments on earlier version of manuscript, although any error are own and should not tarnish reputation of these esteemed persons.

REFERENCES

[1] Tariqul Islam, Md. Saiful Islam, Md. Shajid-Ul-Mahmud, and Md Hossam-E-Haider Comparison of complementary and Kalman filter based data fusion for attitude heading reference system. Proceedings of the 1st International Conference on Mechanical Engineering and Applied Science (ICMEAS 2017).

[2] Yost Labs, 3-Space Sensor Miniature Attitude and Heading Reference System User Manual.

Robust Bayesian Aggregation for Radioactive Source Detection in Urban Scenes

Jack H. Good, Ian Fawaz, Kyle Miller, and Artur Dubrawski *Member, IEEE*

***Abstract*— Detection of potentially hazardous radioactive materials is an important part of the prevention of radiological dispersal devices and accidental exposure to harmful levels of radiation. There are widespread systems in place for detecting such materials passing through stationary radiation portal monitors [23]; these environments are stable and controlled. However, urban environments, in which there is potential for great harm due to dense population, are dynamic, crowded, and widely varied in physical structure and radioactive background, making the source detection problem difficult, especially when mobile sensors are desirable. We propose two models built on the Bayesian Aggregation (BA) framework: one that maximizes, and one that marginalizes, over uncertain variables. These show improved robustness over baseline BA to urban detection challenges such as background radiation variability, positional uncertainty, and dynamic occlusion. These methods can be applied with small gamma-ray sensors mounted on law enforcement and public service vehicles for low-cost and widespread integration.**

I. Introduction

Source detection is determining the presence or absence of radioactive materials of interest, often using gamma-ray spectroscopic data. This problem has applications in medicine [14], industrial safety [8] [15], and national security [7] [32]. This work focuses on the security application of source detection in urban environments, a likely target for attacks with radiological dispersal devices where the cost of failing to detect threats is potentially tremendous. Many detection challenges, including variation in background radiation, uncertain sensor position, dynamic occlusions, moving sources, anisotropic shielding, and nuisance sources, are more pronounced in this busy and dynamic environment. In addition, methods that operate with, small, inexpensive, and mobile sensors are desirable to facilitate the monitoring of large areas and the integration of detection systems with existing infrastructure. It is thus of interest to develop source detection algorithms that support small, mobile sensors and are robust to detection challenges.

We design such algorithms using Bayesian Aggregation (BA), a framework for leveraging evidence from multiple observations [27] that has a history of application in source detection. We build upon this framework to present two novel source detection techniques that show improved robustness to uncertainty: one that maximizes the BA score over uncertain variables, and one that marginalizes over uncertain variables.

Our analysis focuses specifically on our methods' potential to perform under variable background, uncertain position, and dynamic occlusion. We compare against a vanilla BA baseline and an oracle (fully informed) model to gauge performance. We also provide an asymptotic approximation of the maximum improvement of the oracle over a partially aware model, and show empirically that our approaches come close to meeting this bound. Our results generally favor marginalization as the more effective and flexible approach, with the caveat that it is significantly more computationally demanding than the maximization approach.

J. Good, I. Fawaz, K. Miller, and A. Dubrawski were with the Auton Lab, The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA

II. Related Works

This section will outline relevant work in source detection, including both simulations and methods used to further the robustness of detection. We also cover work in Bayesian Aggregation and its use in source detection and elsewhere.

Simulations are a common tool for testing source detection techniques. Urban environments in particular have been the focus of several: the *Half Life 2* game engine was used to explore anisotropic shielding and attenuation through various common urban materials [17], while another project studies the detection of sources on the surface of common urban materials at large standoff distance [6]. The real-time detection of nuclear threats with distributed networks of mobile sensors, e.g. sensors mounted on taxi cabs, has also been simulated [11].

The techniques proposed to improve source detection have been varied. Multiple small, mobile sensors working in conjunction can offer better performance in certain environments than stationary sensors or a single large sensor, while covering a larger area [17] [19] [33]. A Poisson-Clutter Split algorithm was deployed to model background clutter and perform a Generalized Likelihood Ration Test for detection [4]. The problem of low photon count sources, meanwhile, has been addressed using Poisson modeling and Bayesian estimation [28], as well as the G-P detection method for use with large sensors and the List Mode Regression method for use with small sensors [13]. Neutron-based detection methods, in which neutron interaction with various materials causes characteristic gamma-ray emissions, have been deployed extensively [26] [9] [10] [25]. Radiation threats have been detected at long distance using lidar [2] and laser-induced breakdown spectroscopy [6]. Directional detection has been performed with coded-aperture masks [18] [20]. Principle component analysis has been commonly used with spectroscopic data [3] [17] [24] [30] [33]. Censored energy windowing is another common tool [16] [19] [20] [27]. This was extended with Canonical Correction Analysis, and the

background tracked with a Kalman filter rather than using global background [16].

Bayesian aggregation has been applied extensively in source detection to leverage evidence from multiple sensors or observations [20] [27]. It has also been used to building a model of background data, which is then injected with synthetic sources in order to learn models of radiation threats [29]. Outside source detection, BA has been used in the aggregation of ranking functions [5] and of first nonatomic probabilities and second Savagean orderings [21].

III. Methods

A gamma-ray spectroscopic measurement is represented as a vector $x \in \mathbf{N}^m$ of positive integers representing the number of photons collected by the sensor at m different energy levels over a period of time. We can view x as a vector of independent Poisson random variables with parameters $\lambda = B + S$, where B and S are the background and source rate vectors, respectively. The source rate vector is given as $S = \frac{I}{d^2} a \circ s$, where I is the source intensity, d is the distance between the sensor and source, a is a vector of energy-specific attenuation factors resulting from occlusion, and s is the source spectrum with $\sum_j s_j = 1$. For our purposes, we define I to be the expected total photon count when the distance from the sensor to the source is 1.

A. Bayesian Aggregation

The BA score function combines evidence from a set $X = \{x_i\}_{i=1}^n$ of observations and is given as

$$BA(X) = \prod_{i=1}^{n} \frac{P(x_i|\text{source})}{P(x_i|\text{no source})}.$$

This score is compared to a decision threshold τ to predict the presence or absence of a source. To compute the score, we must define a likelihood model $P(x_i|\cdot)$. Previous work [19] has defined $P(x_i|\cdot, I)$ in terms of I, the source intensity, so that this unknown value can be integrated out. We extend this approach by defining a vector $\zeta \in \Omega^n$ of values that capture not only the intensity, but also the distance between the sensor and source and the attenuation due to occlusion at each observation. This allows us to account for these additional uncertainties by defining two aggregate score functions: one that maximizes score over ζ, and one that marginalizes over ζ.

$$BA_{\text{max}}(X) = \max_{\zeta \in \Omega^n} P(\zeta) \prod_{i=1}^{n} \frac{P(x_i|\text{source}, \zeta_i)}{P(x_i|\text{no source}, \zeta_i)}$$

$$BA_{\text{marg}}(X) = \int_{\Omega^n} \cdots \int P(\zeta) \prod_{i=1}^{n} \frac{P(x_i|\text{source}, \zeta_i)}{P(x_i|\text{no source}, \zeta_i)} d\zeta_1 \ldots d\zeta_n$$

In order to make the score computable for many observations, we assume that the ζ_i are independent given I. This allows us to write the score as

$$BA_{\text{max}}(X) = \max_{I \in \mathbf{R}} P(I) \prod_{i=1}^{n} \max_{\zeta_i \in \Omega} \frac{P(\zeta_i|I) P(x_i|\text{source}, \zeta_i)}{P(x_i|\text{no source}, \zeta_i)}$$

$$BA_{\text{marg}}(X) = \int_{\mathbf{R}} P(I) \prod_{i=1}^{n} \int_{\Omega} \frac{P(\zeta_i|I) P(x_i|\text{source}, \zeta_i)}{P(x_i|\text{no source}, \zeta_i)} d\zeta_i \; dI$$

We now require an intensity prior $P(I)$; a prior $P(\zeta_i|I)$, in which we encode any knowledge about the environment gleaned from auxiliary data; and a conditional likelihood model $P(x_i|\cdot, \zeta_i)$.

B. Matched Filter Likelihood Model

Matched filter [31] is the linear model that maximizes the signal to noise ratio for given source template and background. For background covariance Σ computed from training data, it is given as $\text{MF}(x) = s^T \Sigma^{-1} x$. We define a weight vector $w = \Sigma^{-1} s$ such that $\text{MF}(x) = w \cdot x$ to simplify the notation.

Recall that a single observation x is a vector of independent Poisson random variables with parameters $\lambda = B + \frac{I}{d^2} a \circ s$. Since the matched filter score is a weighted sum of Poisson random variables, it is approximated by $MF(x) \sim N(w \cdot \lambda, (w \circ w) \cdot \lambda)$. We define $\zeta = \frac{I \sum_j w_j a_j s_j}{d^2 \sum_j w_j s_j}$ so that $w \cdot \lambda = \mu_b + \zeta \mu_s$ and $(w \circ w) \cdot \lambda$ is approximated by $\sigma_b^2 + \zeta \sigma_s^2$, where

$$\mu_b = w \cdot B \qquad \sigma_b^2 = (w \circ w) \cdot B$$
$$\mu_s = w \cdot s \qquad \sigma_s^2 = (w \circ w) \cdot s$$

This allows us to construct a likelihood model for a single observation x and corresponding scalar ζ as

$$\frac{P(x|\text{source}, \zeta)}{P(x|\text{no source}, \zeta)} = \sqrt{\frac{\sigma_b^2}{\sigma_b^2 + \zeta \sigma_s^2}} \exp\left(\frac{(MF(x) - \mu_b - \zeta \mu_s)^2}{2(\sigma_b^2 + \zeta \sigma_s^2)} - \frac{(MF(x) - \mu_b)^2}{2\sigma_b^2} \right)$$

C. Priors

Assuming there is no knowledge about the intensity of the source, the prior $P(I)$ is really just necessary for the integral to converge; it need not be used at all when maximizing. As for marginalizing, in keeping with previous work [19], we select an upper bound on source intensities we realistically expect to exist and allow $P(I)$ to be uniform from zero to this upper bound.

There is a large degree of flexibility in defining $P(\zeta|I)$ depending on the auxiliary data available. We assume that we have a noisy estimate of the sensor position, as provided by a GPS, from which we compute an estimate $\hat{d}$ of the distance between the sensor and hypothesized fixed-position source. Since we have no evidence regarding occlusion, we estimate $\hat{a}_{ij} = 1$ for all i, j. We now compute $\hat{\zeta}_i = \frac{I \sum_j w_j \hat{a}_{ij} s_j}{\hat{d}_i^2 \sum_j w_j s_j} = \frac{I}{\hat{d}_i^2}$ and apply L1 regularization to the score by letting $P(\zeta_i|I) = \exp(-\frac{\lambda}{I}|\zeta_i - \hat{\zeta}_i|)$ for some regularization strength λ. We divide by I to effectively exclude it from the regularization, since it is given, and prevent it from changing the strength of

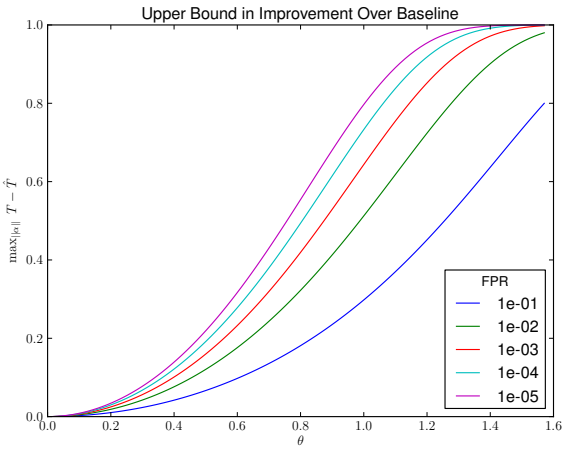


Fig. 1. Upper bound in the improvement of the true positive rate of the oracle over that of the baseline at several fixed false positive rates.

regularization on the estimated terms. We also set a minimum value on the distance estimates $\hat{d}_i$ to prevent extremely large $\hat{\zeta}_i$, which cause very poor detection if the true distance is not actually so small.

D. Asymptotic Approximation of Boundary in Improvement

In examining the efficacy of these models, it is of interest to determine a boundary in the improvement in decision performance that is possible, given perfect information about the data generating process, over an imperfect baseline model. The fully informed model, which we refer to as the oracle, determines the presence of the source optimally with the Bayes' decision rule $\ell(x) > \tau$, where τ is a decision threshold and $\ell(x) = -I + \sum_j x_j \ln(1 + Is_j/B_j)$ is the log-likelihood ratio of source presence over source absence. Previous work [19] derives an approximation for the true positive rate of the oracle; we repeat this analysis in terms of $S = \{S_i\}_{i=1}^n$, the set of true source rate vectors used in the data generating process, $\hat{S} = \{\hat{S}_i\}_{i=1}^n$, the set of estimated source rate vectors used by the model, and B, the background rate vector, which we fix across observations. The approximation is given as

$$T \approx \frac{1}{2} - \frac{1}{2}\text{erf}\left(\gamma - \frac{\sum_{ij} \frac{S_{ij}\hat{S}_{ij}}{B_j}}{\sqrt{2\sum_{ij}\frac{\hat{S}_{ij}^2}{B_j}}}\right),$$

where $\gamma = \text{erf}^{-1}(1 - 2F_0)$, with F_0 the desired false positive rate. For the oracle model, where $\hat{S} = S$, we have $T \approx \frac{1}{2} - \frac{1}{2}\text{erf}(\gamma - \frac{1}{\sqrt{2}}||\alpha||)$, with α the vector defined by $\alpha_i = \sum_j S_{ij}/\sqrt{B_j}$. For the model that estimates the rate vectors, we have $\hat{T} \approx \frac{1}{2} - \frac{1}{2}\text{erf}(\gamma - \frac{1}{\sqrt{2}}\frac{\alpha\cdot\beta}{||\beta||})$, with β the vector defined by $\beta_i = \sum_j \hat{S}_{ij}/\sqrt{B_j}$. We define θ as the angle between α and β; this provides a measure of the difference between the actual and estimated parameters of the data generating process. By rewriting $\hat{T}$ in terms of θ and $||\alpha||$ and maximizing $T - \hat{T}$ over $||\alpha||$, we obtain the maximum possible improvement of the oracle over the baseline as

$$\max_{||\alpha||} \; T - \hat{T} \approx \text{erf}\left(\gamma\frac{1-\cos\theta}{1+\cos\theta}\right).$$

It is unsurprising to find that the boundary increases with θ; the less accurate our parameter estimates, the greater the possibility for improvement by correcting them. A more noteworthy trend is that, as the desired false positive rate decreases, the potential for improvement grows large (see Fig. 1).

IV. Scene Simulation

In order to generate suitably realistic inputs for testing our models, we built a simple simulation of a busy two-lane urban road with parking on both sides. Somewhere on the roadside we place a source near the ground, and on the roof of a car we place a sensor. This car makes several passes while the simulation produces exposure vectors from distance and attenuation information integrated over one-second intervals. These are later paired with a background rate vector, source spectrum, and intensity to generate spectroscopic samples.

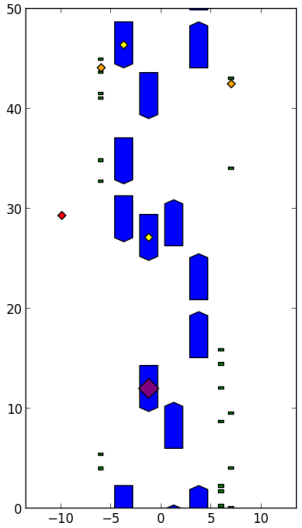


Fig. 2. A top-down, simplified visualization of one frame from our scene simulation. The large rhombus is the sensor, while the many small rhombi are sources.

We include two solids in the scene: pedestrians (green in Fig. 2), each of which is a single cuboid of human size; and cars (blue in Fig. 2), each of which consists of two cuboids for the body, one for the engine block, and one to four for the occupants, of which there is always one in the driver position. We use ray tracing from the sensor to the source to identify the gamma rays' interaction with each object. We thus compute the exposure vector $\eta \in [0,1]^m$ according to the inverse square law and exponential attenuation law as

$$\eta_j = \frac{1}{d^2}\exp\left(-\sum_k \mu_{kj} z_k\right),$$

where μ_{kj} is the attenuation coefficient at the jth energy level for, and z_k the distance passed through, the kth cuboid. We compute the μ_k vector by interpolating mass attenuation coefficients from NIST Physical Measurement Laboratory databases [12] to match the energy levels in our background and spectral data; we treat pedestrians and vehicle occupants as solid water, car bodies as a hollow surface of iron, and engine blocks as solid iron.

To introduce variety, we randomize the parked cars and idle pedestrians at the initialization of each scene. For each sensor pass, we randomly select a speed for each road lane and parameters that determine how densely the road is filled with passing cars and the sidewalk with passing pedestrians.

We add additional sources and attach them to objects, as if they were in a pedestrian's backpack or in the back seat of a car. We place two on randomly selected stationary objects when the scene is initialized and assign a probability that, when an object is spawned, it will have a source attached. These additional exposure data are generated with the intention to analyze the effect of nuisance sources; however, this analysis has not yet been performed.

V. Results

In our analysis we present several scatterplots showing improvement over baseline BA versus θ. Each point was generated by selecting a single set of exposure values from the scene simulation, a background rate vector drawn at random from empirical data, a random source intensity, and a source spectrum. The background data is from the RadMAP Project [1] [22] and was collected by a detection vehicle on the roads of West Berkeley, Downtown Berkeley, South Side, and Solano, California on May 1, 2012. The source spectrum is shielded SNM provided by our collaborators at Lawrence Livermore National Laboratory. When considering positional uncertainty, we apply Gaussian noise to the horizontal coordinates of the sensor with a mean of zero and standard deviation of three meters, in order to approximate GPS error in urban scenes. With all these scene parameters fixed, we sample several thousand matched filter scores from a normal distribution[1], with and without source injected, and apply each model[2] to obtain BA scores. We then determine the decision threshold for each point that produces the desired false positive rate and use it to obtain the relevant true positive rate. Since the practical application of source detection requires extremely low false positive rate, we perform all our analyses at a target FPR of 10^{-3}. Unless otherwise stated, all results were obtained under all three of the target sources of uncertainty.

A. Evaluation of Boundary in Improvement

Our derivation of the maximum improvement over baseline BA assumes that the background rate vector is perfectly known; however, by showing empirically the improvement of the oracle over the baseline with both certain and uncertain background (Fig. 3), we see that the bound still holds. Thus, in several of the following figures, we show the theoretical bound in lieu of the oracle results.

[1]We sample matched filter scores instead of spectroscopic observations to allow completion of many thousands of tests in reasonable time. See section III-B for details of this approximation.

[2]Also in the interest of simulation time, we approximate the integrals in the marginalization model with the trapezoid rule on the range $[0, 2\hat{\zeta}]$. This is why we are able to show regularization penalty 0 in Fig. 4 for an integral that would not converge if evaluated over its full domain. In addition, we allow all models, including the baseline, to know the true source intensity so that we need not evaluate the expensive double integral in the marginalization model.

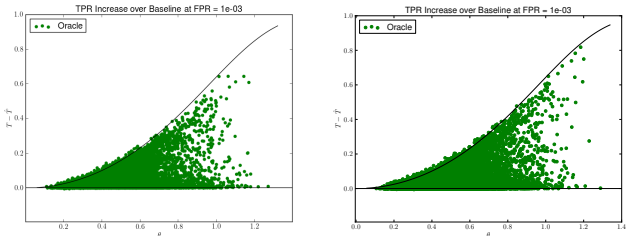


Fig. 3. Oracle improvement over the baseline with certain (left) and uncertain (right) background radiation.

B. Regularization Strength

In examining the effect of L1 regularization prior $P(\zeta|I) \propto \exp(-\frac{\lambda}{I}|\zeta - \hat{\zeta}|)$, we provide results for each model under several values for the regularization penalty λ (Fig. 4). The general trend is that increasing the penalty largely recovers losses when θ is small, i.e., the estimated parameterization of reality is close to ground truth; meanwhile it slightly decreases gains when θ is large, i.e., the estimated parameterization of reality is far from the ground truth. The losses are significantly more pronounced in the maximization model and occur at all but the largest θ, making it more sensitive to regularization tuning.

In all results following, we use penalties 512 and 256 for maximization and marginalization, respectively. These were selected for being the lowest penalties from Fig. 4 that show little apparent loss compared to the baseline.

C. Isolated Sources of Uncertainty

Fig. 5 shows performance of both models when all but a single source of uncertainty are fully known. Uncertainty in background is not shown because, in its isolation, we find no difference between the performance of the our models and of the baseline. This is unsurprising, since θ is always very close to zero when only the background is uncertain. The reduced impact of background variation can likely be credited to the fact that both our theoretical analysis and our experiments allow the decision threshold to be selected individually for each scene.

When positional uncertainty is isolated, we see that both models perform particularly well; the frequency and magnitude of losses versus the baseline are small, and performance comes close to the theoretical bound.

When occlusions are isolated, since our estimation $\hat{\zeta}$ assumes there is no occlusion present, it is not possible that $\zeta > \hat{\zeta}$, so we adjust our priors accordingly. The outcome of these tests is more nuanced. The θ from our simulated scenes tend to cluster on the low end, likely because the relatively fast-moving sensor causes occlusions to be fleeting and thus more uniform across observations; in these clusters we see that there is notable potential for loss versus the baseline for both models. However, when θ is not so small, we do see greater gains and less significant losses, though the exact behavior is hard to characterize due to the scarcity of such cases. It is clear, however, that these models handle occlusion

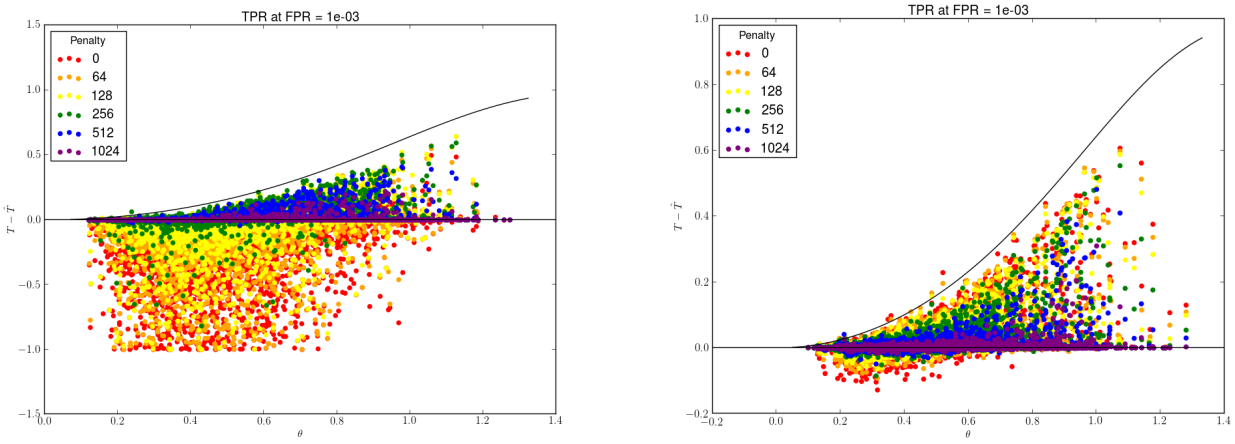


Fig. 4. Performance of maximization (left) and marginalization (right) approaches under various L1 regularization strengths.

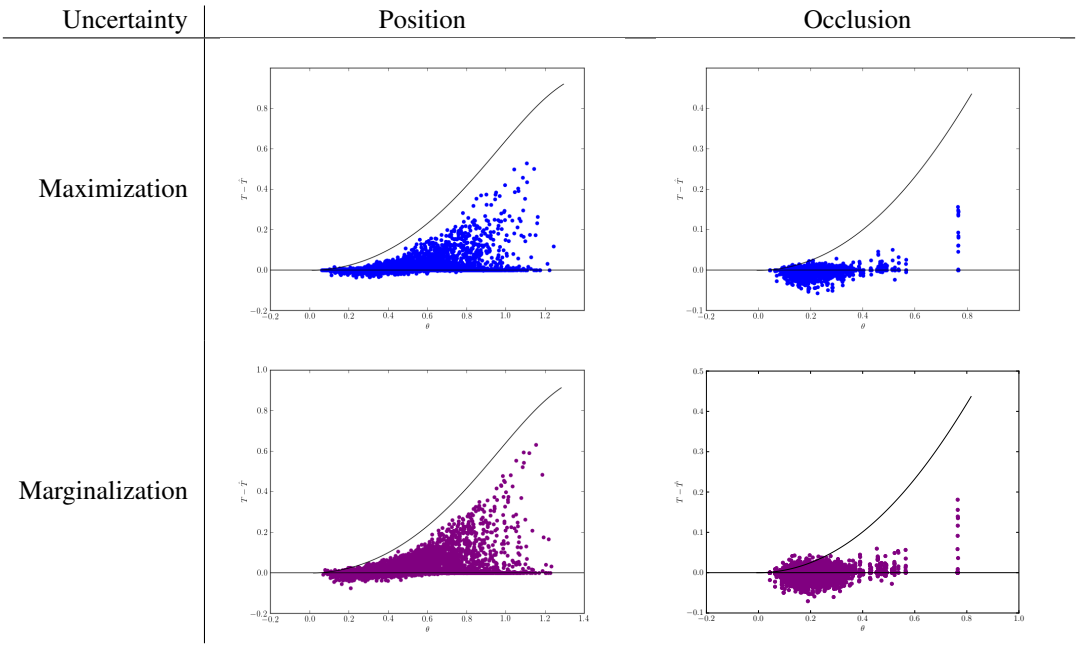


Fig. 5. Performance of each model with sources of uncertainty in isolation. Note the difference in scale.

in isolation less effectively than positional uncertainty with the selected regularization.

D. Comparison of the Two Models

While both models show true positive rates close to the the theoretical bound, and thus are capable of near-oracle performance, each has comparative advantages and disadvantages. Marginalization offers greater robustness to regularization tuning (see section V-B); with the selected penalties, it also frequently shows significantly better performance on individual scenes than maximization (Fig. 6). This may be a result of the lower penalty needed to avoid losses over the baseline; a lower penalty leads to greater flexibility to adapt to uncertainty when θ is large. Maximization, however, offers fast computation, since it is possible to derive an expression for the maximum score over ζ_i, whereas it is necessary to approximate integrals for the marginalization approach. Fast computation offers a significant advantage when the hypothesis space is large, a common problem when one wishes to consider several combinations of source position, composition, and shielding.

VI. Conclusion

Both our models demonstrate improved robustness to uncertainty over baseline Bayesian Aggregation. In many cases they demonstrate near-oracle performance and, with a tuned L1 regularization, avoid losses at low levels of uncertainty. Additionally, they offer the flexibility to incorporate auxiliary data with minimal alteration to the model, simply by adjusting the prior $P(\zeta_i|I)$.

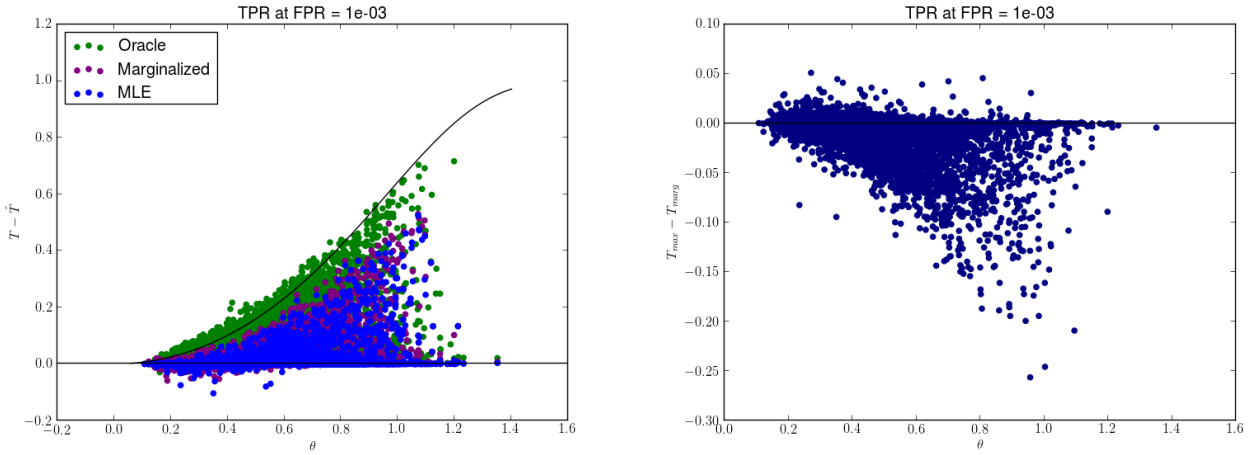


Fig. 6. Both models compared against the baseline (left) and directly against each other (right).

However, our models suffer from the restrictive assumption that the ζ_i are independent given I. This complicates any kind of regularization across observations, which could be used to model occlusions[3] [27] and account for noise in sequential position estimates. Additionally, the marginalization approach can be slow to execute depending on how the integrals are approximated. Because of the large number of trials required to compute true positive rates at a low false positive rate with high confidence, we approximated integrals on a finite domain with the trapezoid rule; however, we expect that in practical application, adaptive methods can be used to approximate integrals over the entire domain in reasonable time, unless the hypothesis space is exceptionally large (which may be the case when considering many combinations of source position, composition, and shielding). Maximization, likewise, is vulnerable to more complex priors, which can can cause it to be difficult or impossible to solve for the maximized score, thus necessitating a numerical maximization and sacrificing the speed of this approach.

If speed is not an obstacle, marginalization is the preferable approach due to its lesser potential for loss over the baseline, its reduced sensitivity to regularization tuning, and its capacity to yield a comparatively higher true positive rate in many cases where θ is large.

A. Limitations and Future Work

Perhaps the greatest limitation of this work is its reliance on simulated data for the evaluation of our models. Though we attempt to adhere to the relevant physics and important aspects of urban source detection, our simulation is ultimately quite simple, and the analysis would benefit from testing involving real sensors and sources in an urban setting. Additionally, due to our focus on urban source detection with vehicle-mounted sensors, we consider only one scenario, i.e., a vehicle-mounted sensor making several passes by a stationary roadside source. While we expect this to be the most common case in urban source detection, it may not generalize well to all real-world scenarios.

Another limitation of our testing is that, in the interest of representing our empirical results against the approximated theoretical bound, we select a decision threshold individually for each test according to a set of scores that were all generated with the same scene parameters, including background rate vector, source intensity, position, occlusion, etc. While this does show our models' potential for improvement over baseline BA, further testing is necessary to show how they will perform in real-world applications, where decision thresholds can not be custom-tuned to every possible scene.

Future work on this project will involve testing the model under the remaining sources of uncertainty, namely moving sources, nuisance sources, and anisotropic shielding. Adaptations to the models and theoretical analysis may be made to account for nuisance sources, which are not yet modeled explicitly.

Work is also underway to integrate these methods with video cameras and computer vision techniques. This can provide basis for moving source hypotheses through object tracking, reduce the hypothesis space in stationary-source problems by considering only locations with suspect objects, and improve priors by ruling out occlusion when there is line of sight to the suspect.

[3]In our simulation we see that occlusion is indeed largely independent across observations, likely because of the speed of the sensor causes occlusions to be fleeting; however, the case may be different for slow-moving or stationary sensors.

VII. Acknowledgments

Jack Good is funded by the Carnegie Mellon University Robotics Institute REU Site (NSF Grant #1659774). He thanks the Robotics Institute Summer Scholars for making his contribution to this work possible. He and Ian Fawaz thank Kyle Miller and Artur Dubrawski for their mentorship.

References

[1] Mark S. Bandstra, Timothy J. Aucott, Erik Brubaker, Daniel H. Chivers, Reynold J. Cooper, Joseph C. Curtis, John R. Davis, Tenzing H. Joshi, John Kua, Ross Meyer, Victor Negut, Michael Quinlan,

Brian J. Quiter, Shreyas Srinivasan, Avideh Zakhor, Richard Zhang, and Kai Vetter. Radmap: The radiological multi-sensor analysis platform. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 840:59 – 68, 2016.

[2] Norman P. Barnes. Challenges for current spectroscopy: detection of security threats. In Baldassare Di Bartolo and Ottavio Forte, editors, *Advances in Spectroscopy for Lasers and Sensing*, pages 63–72, Dordrecht, 2006. Springer Netherlands.

[3] D. Boardman, M. Reinhard, and A. Flynn. Principal component analysis of gamma-ray spectra for radiation portal monitors. *IEEE Transactions on Nuclear Science*, 59(1):154–160, Feb 2012.

[4] B. R. Cosofret, K. N. Shokhirev, P. A. Mulhall, D. Payne, B. Harris, E. Arsenault, and R. Moro. Utilization of advanced clutter suppression algorithms for improved spectroscopic portal capability against radionuclide threats. In *2013 IEEE International Conference on Technologies for Homeland Security (HST)*, pages 618–622, Nov 2013.

[5] Ke Deng, Simeng Han, Kate J. Li, and Jun S. Liu. Bayesian aggregation of order-based rank data. *Journal of the American Statistical Association*, 109(507):1023–1039, 2014.

[6] I. Gaona, J. Serrano, J. Moros, and J.J. Laserna. Evaluation of laser-induced breakdown spectroscopy analysis potential for addressing radiological threats from a distance. *Spectrochimica Acta Part B: Atomic Spectroscopy*, 96:12 – 20, 2014.

[7] B. D. Geelhood, J. H. Ely, R. R. Hansen, R. T. Kouzes, J. E. Schweppe, and R. A. Warner. Overview of portal monitoring at border crossings. In *2003 IEEE Nuclear Science Symposium. Conference Record (IEEE Cat. No.03CH37515)*, volume 1, pages 513–517 Vol.1, Oct 2003.

[8] L. Giusti. A review of waste management practices and their impact on human health. *Waste Management*, 29(8):2227 – 2239, 2009.

[9] Tsahi Gozani. The role of neutron based inspection techniques in the post 9/11/01 era. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 213:460 – 463, 2004. 5th Topical Meeting on Industrial Radiation and Radioisotope Measurement Applications.

[10] Tsahi Gozani and Dan Strellis. Advances in neutron based bulk explosive detection. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 261(1):311 – 315, 2007. The Application of Accelerators in Research and Industry.

[11] Dorit S. Hochbaum and Barak Fishbain. Nuclear threat detection with mobile distributed sensor networks. *Annals of Operations Research*, 187(1):45–63, Jul 2011.

[12] J. H. Hubbell and S. M. Seltzer. X-ray mass attenuation coefficients. https://www.nist.gov/pml/x-ray-mass-attenuation-coefficients. Accessed: 2017-07-23.

[13] Jay Jin. Detection of sources of harmful radiation using portable sensors. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, 2016.

[14] F. F. (Russ) Knapp and Ashutosh Dash. *Introduction: Radiopharmaceuticals Play an Important Role in Both Diagnostic and Therapeutic Nuclear Medicine*, pages 3–23. Springer India, New Delhi, 2016.

[15] K. Krauskopf. *Radioactive Waste Disposal and Geology*. Topics in the Earth Sciences. Springer Netherlands, 2013.

[16] Eric Lei. Robust detection of radiation threat. 2017.

[17] Annie Liu. Simulation and implementation of distributed sensor network for radiation detection. Master's thesis, California Institute of Technology, Pasadena, CA, 2010.

[18] J. L. Matteson, M. A. Capote, R. T. Skelton, G. J. Batinica, E. Stephan, R. E. Rothschild, G. Huszar, T. Gasaway, and M. R. Pelling. A directional gamma radiation spectrometer based on pixelated czt arrays and coded mask apertures. In *2006 IEEE Nuclear Science Symposium Conference Record*, volume 1, pages 87–92, Oct 2006.

[19] K. Miller and A. Dubrawski. Gamma-ray source detection with small sensors. *IEEE Transactions on Nuclear Science*, 65(4):1047–1058, April 2018.

[20] Kyle Miller, Peter Huggins, Simon Labov, Karl Nelson, and Artur Dubrawski. Evaluation of coded aperture radiation detectors using a bayesian approach. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 839:29 – 38, 2016.

[21] Philippe Mongin. Consistent bayesian aggregation. *Journal of Economic Theory*, 66(2):313 – 351, 1995.

[22] Brian J. Quiter, Lavanya Ramakrishnan, and Mark S. Bandstra. Grdc: a collaborative framework for radiological background. Technical report, Lawrence Berkeley National Laboratory, 12 2015.

[23] R. C. Runkle, M. F. Tardiff, K. K. Anderson, D. K. Carlson, and L. E. Smith. Analysis of spectroscopic radiation portal monitor data using principal components analysis. *IEEE Transactions on Nuclear Science*, 53(3):1418–1423, June 2006.

[24] R. C. Runkle, M. F. Tardiff, K. K. Anderson, D. K. Carlson, and L. E. Smith. Analysis of spectroscopic radiation portal monitor data using principal components analysis. *IEEE Transactions on Nuclear Science*, 53(3):1418–1423, June 2006.

[25] Robert C. Runkle, Mitchell J Myjak, Michael T. Batdorf, Ryan S. Bowler, Scott D. Kiff, Scott J. Morris, Crystal A. Mullen, John S. Rohrer, and Lindsay C. Todd. Unattended sensors for nuclear threat detection. volume 6954, pages 6954 – 6954 – 8, 2008.

[26] Dan A. Strellis, Mashal Elsalim, and Tsahi Gozani. Explosives (and other threats) detection using pulsed neutron interrogation and optimized detectors. volume 8017, pages 8017 – 8017 – 6, 2011.

[27] Prateek Tandon. *Bayesian Aggregation of Evidence for Detection and Characterization of Patterns in Multiple Noisy Observations*. PhD thesis, Carnegie Mellon University, 7 2015.

[28] Prateek Tandon, Peter Huggins, Artur Dubrawski, Karl Nelson, and Simon Labov. Poisson modeling and bayesian estimation of low photon count signal and noise components.

[29] Prateek Tandon, Peter Huggins, Rob Maclachlan, Artur Dubrawski, Karl Nelson, and Simon Labov. Detection of radioactive sources in urban scenes using bayesian aggregation of data from mobile spectrometers. *Inf. Syst.*, 57(C):195–206, April 2016.

[30] Mark F. Tardiff, Robert C. Runkle, K. K. Anderson, and L. E. Smith. Anomaly detection in gamma-ray vehicle spectra with principal components analysis and mahalanobis distances. 1 2006.

[31] G. Turin. An introduction to matched filters. *IRE Transactions on Information Theory*, 6(3):311–329, June 1960.

[32] Lawrence M. Wein, Alex H. Wilkins, Manas Baveja, and Stephen E. Flynn. Preventing the importation of illicit nuclear materials in shipping containers. *Risk Analysis*, 26(5):1377–1393.

[33] Jifu Zhao. Radiation source detection from mobile sensor networks using principal component analysis. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2016.

Online Learning for Obstacle Avoidance with Powered Transfemoral Prostheses

Max Gordon[1], Nitish Thatte[2], and Hartmut Geyer[2]

***Abstract*— Avoiding obstacles is a significant challenge for amputees using transfemoral prosthetic limbs due to the lack of knee control available to them. Powered prostheses may make obstacle avoidance easier than passive prostheses due to their ability to add energy to the amputees stride. Stumble recovery systems have been implemented on such powered prosthetic limbs that improve recovery from a failed obstacle avoidance attempt. However, these systems only aid in recovery after an obstacle has disrupted the users gait, and do not allow the user to more easily avoid the initial obstacle than an unaltered powered prosthetic limb. We designed an adaptive system using kinematic data from the prosthetic limb to detect obstacle avoidance responses in the prosthetic limbs user before the portion of the stride critical to obstacle avoidance. We then used this system to alter the prosthetic limbs planned swing trajectory to help avoid the obstacle. The system used a regression model to predict the user's desired level of obstacle response. We validated our system by comparing obstacle avoidance success rates with and without the obstacle avoidance response prediction.**

I. INTRODUCTION

Successfully navigating around and avoiding obstacles on the ground is an important task in everyday life, and is necessary to maintain basic safety while performing a large number of tasks. Amputees have significant difficulties in safely navigating their environment, with an extremely high annual fall rate [1]. This is made particularly challenging by current prosthesis technology. In particular, current transfemoral prostheses, used for patients with above-knee amputations are capable of more closely reproducing normal human gait [2], [3]. This provides little to no control of the knee joint, making it challenging to gain the necessary height above the ground to clear some obstacles. Powered prostheses can improve obstacle avoidance abilities by providing more predictable knee flexion and adding power to the user's gait cycle, but the underlying problem of lack of control remains. Existing research focused on improving an amputee's ability to navigate obstacles is largely integrated into stumble recovery systems [4], providing assistance after an obstacle has caused a disruption. There has been work in online learning systems for powered transfemoral prostheses for the problem of gait classification, including the integration of electromyograph data. We propose an obstacle avoidance system for powered prostheses using early-swing measurements of the mechanical state of the prosthesis and position and velocity of the user's residual limb to determine whether an obstacle avoidance attempt is in progress. After detection, the system alters the behavior of the prosthesis to aid in the avoidance of the obstacle. This system is made robust through online learning using a forward-backward classifier feedback system to use information from later in the step to refine the predictive model. We also developed a regression system to predict the appropriate obstacle avoidance trajectory for the user's avoidance response using the obstacle avoidance backward classifier margins.

*This work was supported by the National Science Foundation under Grant No. 1734559.

[1]Max Gordon mjgordo3@ncsu.edu is with the Department of Electrical and Computer Engineering, North Carolina State University, 890 Oval Dr, Raleigh, NC 27606, USA

[2]Nitish Thatte nitisht@cs.cmu.edu and Hartmut Geyer hgeyer@cs.cmu.edu are with the Robotics Institute, Carnegie Mellon University, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213, USA

II. METHODS

A. Prosthesis

We used a powered knee and ankle prosthesis and minimum jerk swing controller. The mechanical features we used for our online learning system were calculated from data provided by the hip IMU unit of the prosthesis. The prosthesis is shown in Figure 1.

Fig. 1. Powered Transfemoral Prosthesis

B. Feature Extraction

The same features were used for the forward classification, backward classification, and trajectory regression, consisting of the mean, standard deviation, minimum value, and maximum value of 5 physical signals in the feature

extraction window. For forward classification and regression we used window beginning 210ms before prosthesis toe-off and ending 90ms after, while for the backward classification we used a window consisting of the entire swing phase between toe-off and heelstrike.

C. Forward-Backward Classifier

To adjust the swing trajectory in time to avoid an obstacle, an obstacle avoidance attempt must be detected early in the swing phase. We used a linear support vector machine classifier and features calculated in the early swing phase to detect obstacle avoidance attempts. As user behavior changes over time in response to changes in the environment and prosthesis obstacle avoidance, we retrain this forward support vector machine every 10 steps using labels from the backward classifier. The backward classifier is another linear support vector machine trained once for each user that uses features extracted from the entire swing phase to label a step as an avoidance attempt after the fact. Figure 2 provides an overview of this system.

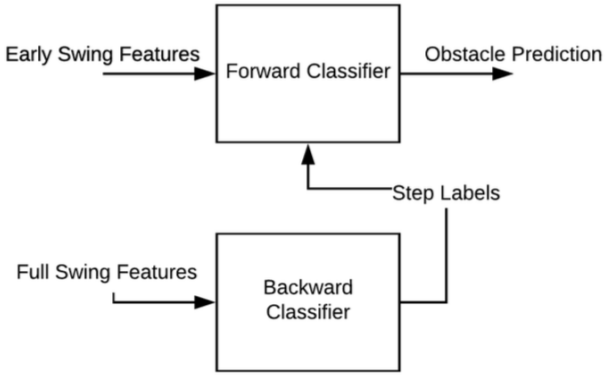


Fig. 2. Forward-Backward Classifier Overview

D. Target Knee Angle Regression

A prosthesis user will not always encounter obstacles of the same height. As the obstacle avoidance response can be disruptive to the prosthesis user, it is desirable to give the user control over the magnitude of the prosthesis response. We accomplished this by using the normalized backward classifier scores of detected obstacle avoidances as a metric for the difficulty to avoid the obstacle. The knee angle regression was performed by a linear support vector machine, retrained every 10 steps to target its own output for each step modified by a proportional decay term to prevent runaway increase and the class scores of each step. Before use, the class scores were normalized so that scores above the 10th percentile produced positive values and so that scores in the 90th percentile or above would cause an angle to be unaffected by decay at maximum flexion. This resulted in larger commanded angles for pronounced obstacle avoidance behavior by user.

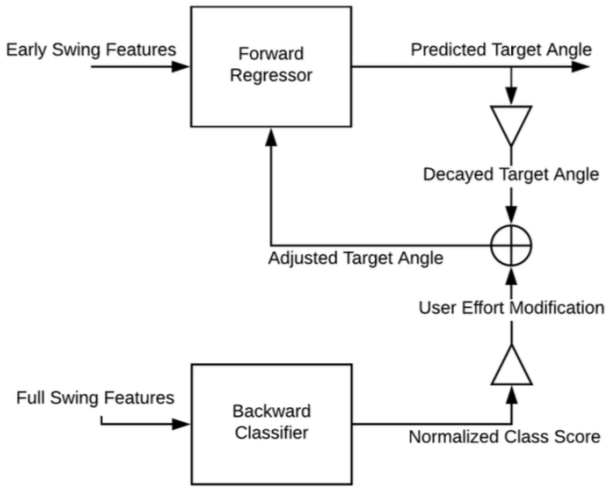


Fig. 3. Knee Angle Regression Overview

E. Avoidance Trajectory Planning

We designed a bang-bang controller to provide maximum obstacle clearance while staying within the prosthesis joint limits. When an obstacle avoidance attempt is detected, the minimum jerk swing trajectory produced by the normal swing controller is replaced with the obstacle avoidance trajectory. The target knee angle regression is used to determine the appropriate peak angle for the knee trajectory, while the ankle trajectory is planned to provide the same proportion of its maximum possible clearance as the knee trajectory. We made this simplification to speed the convergence of the angle regression. Examples showing the minimum jerk swing trajectory and obstacle avoidance trajectories planned for medium and high difficulty obstacles are given in Figure 4

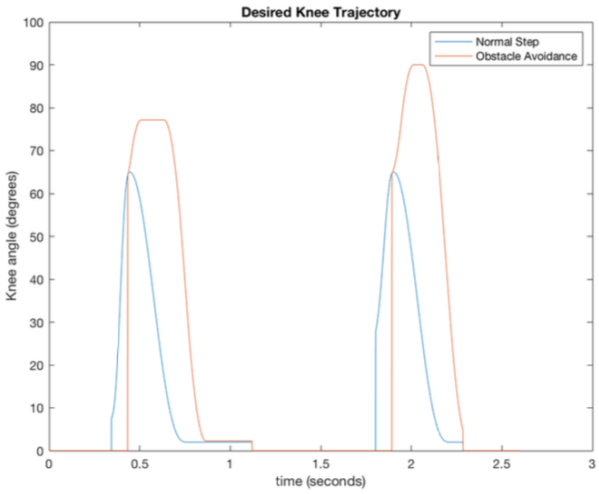


Fig. 4. Trajectory Planning Comparison

III. RESULTS

A. Obstacle Avoidance Success Rate

We tested our obstacle avoidance system by placing obstacles of varying size in front of a subject walking using the prosthesis. Experiments were also conducted using the base minimum jerk swing controller to provide a point of comparison for the obstacle avoidance capabilities of the system. The results of this testing are shown in Table I and show that our online learning system significantly improves obstacle avoidance success rates.

TABLE I

OBSTACLE AVOIDANCE SUCCESS RATES

Controller	Success Rate
Minimum Jerk	37%
Adaptive Bang-Bang	89%

B. Knee Angle Regression Behavior

We also examined the behavior of the knee angle regression to determine its necessity to the obstacle avoidance system. As increased knee flexion is disruptive to the user, we would like to use the minimum possible response while still guaranteeing knee clearance. As shown in Figure 5, our system is able to ensure that high classification score steps, associated with high user effort, obtain larger angle regressions.

IV. DISCUSSION

We developed an online learning system to assist users of powered transfemoral prostheses in obstacle avoidance. Our system makes use of sensors already integrated into most existing powered prostheses and provides the user volitional control of the obstacle avoidance response. Our system requires a brief supervised training period for each user, but otherwise acts without human oversight.

In the future, we look to expand upon this work by incorporating electromyographic data into the obstacle avoidance system. Other potential avenues for future work include integration with LIDAR-based localization and mapping to better plan trajectories around specific obstacles.

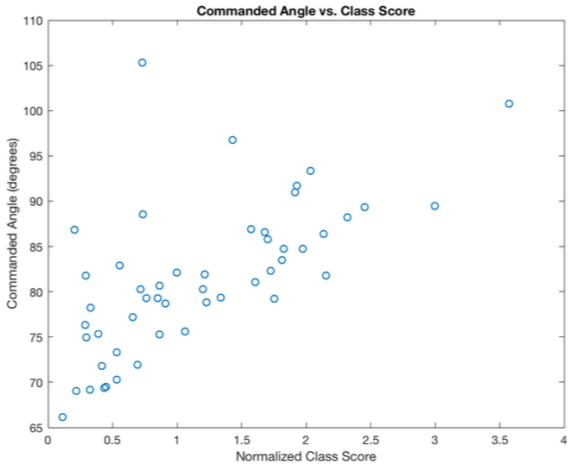


Fig. 5. Backward Classifier Score and Regression Angle

V. ACKNOWLEDGMENT

Max would like to thank the National Science Foundation Robotics Institute REU for funding this work. He would also like to thank Dr. Hartmut Geyer and Nitish Thatte for their guidance in this project, as well as Rachel Burcin, John Dolan and the Robotics Institute Summer Scholars team for their support.

REFERENCES

[1] J. Kulkarni, S. Wright, C. Toole, J. Morris, and R. Hirons, "Falls in Patients with Lower Limb Amputations: Prevalence and Contributing Factors," *Physiotherapy*, vol. 82, pp. 130–136, Feb. 1996.

[2] F. Sup, A. Bohara, and M. Goldfarb, "Design and Control of a Powered Transfemoral Prosthesis," *The International Journal of Robotics Research*, vol. 27, pp. 263–273, Feb. 2008.

[3] H. Geyer, N. Thatte, and H. Duan, "Toward Balance Recovery with Active Leg Prostheses Using Neuromuscular Model Control," in *Converging Clinical and Engineering Research on Neurorehabilitation II* (J. Ibez, J. Gonzlez-Vargas, J. M. Azorn, M. Akay, and J. L. Pons, eds.), vol. 15, pp. 649–652, Cham: Springer International Publishing, 2017.

[4] B. E. Lawson, H. A. Varol, F. Sup, and M. Goldfarb, "Stumble detection and classification for an intelligent transfemoral prosthesis," pp. 511–514, IEEE, Aug. 2010.

Two-State Control Structure for 2 DOF Aerial Manipulator on 6 DOF Hexarotor*

Enrique Maytorena Guemez[1] and Sebastian Scherer[2]

***Abstract*—Advances in robotics are being madde all around the world for every possible scenario. One of the most important and prominent branches of robotics refers to Unmanned Autonomous Vehicles (UAVs). These UAVs can be built and programmed for either research or commercial purposes. Nowadays, most UAVs are more focused on the sensing stage of robotics, rather than actuation. Nevertheless, there are multiple applications where actuating UAVs would be useful. This paper focuses on a two state control structure of a robotic arm on an aerial manipulator self-designed UAV for civil inspection and general purposes. The robotic arm control uses a Proportional-Integral-Derivative (PID) algorithm to fulfill two major tasks: setting the end effector to a position and maintainting that same position. For the first task, a triangulation algorithm is used, while the second uses data from the UAVs Pixhawk. These algorithms were able to reach and mantain the desired position 97.47% of the run.**

I. INTRODUCTION

Advances in robotics are being madde all around the world for every possible scenario. One of the most important and prominent branches of robotics refers to Unmanned Autonomous Vehicles (UAVs). These UAVs can be built and programmed for either research or commercial purposes. Nowadays, most UAVs are more focused on the sensing stage of robotics, rather than actuation. Nevertheless, there are multiple applications where actuating UAVs would be useful. Some of these applications include but are not limited to disaster response UAVs, military UAVs and civil engineering support UAVs. MBZIRC is one of the most important robotics competitions around the world using UAVs, enclosing development for the applications listed above. With the objective of competing on the MBZIRC 2020 event and developing a civil inspection support drone, a 6 DOF Hexarotor (Fig.1) was designed and built at Carnegie Mellon Universitie's Airlab. For fulfilling these two purposes, the drone has an aerial manipulator, which is a 3 DOF self-designed robotic parallel arm (Fig.2). The purpose of this paper is to describe the control structure used for this aerial manipulator, it's main capabilities being to point and lock the end-effector.

II. THE TWO-STATE CONTROL STRUCTURE

The control structure was divided into two states for fulfilling the manipulator's main purposes. The first state is the Free State, destined to point the end-effector to any desired point within reach. The second one is the Locked State, destined to maintain the end-efector at an absolute position after setting it.

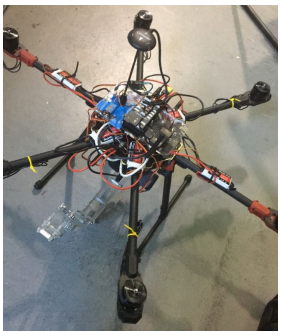


Fig. 1. Figure 1. Picture of the Hexarotor with prototype Aerial Manipulator.

Fig. 2. Figure 2. CAD Design of Aerial Manipulator rendering.

A. Free State

The Free State control is an closed loop control structure that uses a numerical method of triangulation based on the bisection method. The control structure consists on receiving a desired position signal and then modifying the actual position through the triangulation. The method requires to have a mesh with at least the 8 most important reach endpoints of the arm and the angles at which the two extension joints must be set (Fig. 3). Then, we do the following steps:

1) Create a triangle through the following vertex criteria: A is the current end-effector position, B is the closest endpoint to desired position and C is the closest endpoint to AB midpoint on desired position's direction.
2) Find the triangle's centroid and set the servo's to that position.
3) Repeat the loop.

*Carnegie Mellon University and Monterrey Institute of Higher Education and Technology

[1]E. Maytorena is with Faculty of Engineering and Science, Computer Department, Monterrey Institute of Higher Education, and Technology, Mexico.

[2]S. Scherer is with the Robotics Institute, Carnegie Mellon University, The Airlab, Pittsburgh USA

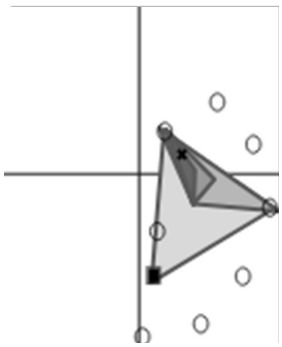


Fig. 3. Figure 3. Representation of triangulation method. Square is the start point, circles represent the endpoint mesh and X is the desired position.

B. Locked State

The main objective of the Locked State is to compensate for the UAV's unplanned displacements while using the end-effector for a task that requires stability. The control system works to a closed loop control between the Pixhawk and the Arm, since the displacement is compensated both by the UAV's general control and the manipulator's. To achieve this, the angle between the Y and Z axis displacements registered by the Pixhawk's IMU is calculated. Then, the position at which each joint has to be calculated and sent to the servos. For the calculation of the angles, the following set of equations is used:

$$\theta_v = 180 - A - \alpha - \sin(L_b \sin(\beta + B)/L_v)$$

Where A = Main arm's initial position, α = Main arm's displacement, B = Second arm's initial position, β = Second arm's displacement, L_b = Second arm's length and L_v = Displacement vector length.

III. RESULTS

Since both of the methods have different algorithms, the results obtained also need to be classified between this two, through different tests representative of the method's purpose fulfillment.

A. Free State Method

The method was ran with 4 distinct max-iteration quantities, 10.000 times each. Table I shows the success and error rates of those experiments:

TABLE I
TABLE REPRESENTING THE DATA OBTAINED THROUGH 10.000 REPETITIONS OF THE NUMERIC METHOD.

Method Iterations	Success Rate	Average Error
4	83.78%	10.53%
12	94.23%	5.72%
36	94.94%	3.96%
360	97.47%	4.78 %

B. Locked State Method

For this method's evaluation, the algorithm was tested with an IMU simulation that generated random displacements and checked whether the arm was or was not able to reach that point before the Pixhawk's control was able to fully react. The manipulator simulation was able to react to displacements from 20 to 65 mm 92.76% of the runs, out of 10,000 runs.

IV. CONCLUSIONS

Tilting the motors of a UAV to achieve 6 DOF greatly simplifies the design and control of an aerial manipulator. With just 2 DOF on one and by synchronizing a drone's IMU with the manipulators position control, it is possible to point and lock the end-effector through relatively simple mathematical computations. The impact this milestone has on UAVs present and future is the possibility to use drones for various actuating next. Nevertheless, while a 2 DOF manipulator may work for a 6 DOF hexarotor and point-lock operations, more DOF are needed for more complex tasks. What comes next is to escalate these methods to higher DOF systems while maintaining or increasing the methods simplicity and effectiveness. The next step is to be able to add even more degrees of freedom to the manipulator and it's control, to be able to answer quicker and more efficiently to the displacements, as well as to reach more complex end-effector positions. Also, to let the manipulator have it's own sensor set for not needing to rely on the Pixhawk for all the State Estimation process would be an important addition.

ACKNOWLEDGMENT

The Airlab Aerial Manipulation Mentors: Sebastian Scherer, Professor Oliver Kroemer, Hu Yaohu, Weikun Zhen.

The Airlab Aerial Manipulation Partners: Nawaf Alotabi, Lucas Nogueira, Pinxu Ren.

RISS Program Coordinators: John Dolan, Rachel Burcin, Ziqi Guo.

REFERENCES

[1] H. Mehmood, A Daisy-Chain Control Design for a Multirotor UAV with Direct Force Capabilities, AIAA SciTech Forum, Jan. 2017.
[2] D. Bershadsky, Electric Multirotor Propulsion System Sizing for Performance Prediction and Design Optimization), unpublished.
[3] H. Mehmood, A Maneuverability Analysis of a Novel Hexarotor UAV Concept, 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Jun. 2016.
[4] S. Haivaland, Development of a 500 gram Vision-based Autonomous Quadrotor Vehicle Capable of Indoor Navigation, unpublished.
[5] A. Suarez, Design of an Anthropomorphic Lightweight Dual Arm for Aerial Manipulation, IEEEAcces, accepted for publishing.
[6] M. Jun Kim, A Stabilizing Controller for Regulation of UAV With Manipulator, IEEE Robotics and automation Letters, Vol. 3, Jul. 2018.
[7] A.E. Jimenez-Cano, Aerial Manipulator with a Compliant Arm for Bridge Inspection, ICUAS, Jun. 2017.
[8] Damic, V; Cohodar, M. and Tvrtkovic, M. (2016). Inverse Dynamic Analysis of Hobby Robot uArm by Matlab/Simulink, Proceedings of the 27th DAAAM International Symposium, pp.0095-0101, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-08-2, ISSN 1726-9679, Vienna, Austria.
[9] S. Kim, Aerial Manipulation Using a Quadrotor with a Two DOF Robotic Arm, IROS, Nov. 2013.

A Modular, Multi-Modal Tactile Sensor Demonstrated during Object Grasping

Quintessa Guengerich[1], Eric Markvicka[2], and Carmel Majidi[3]

***Abstract*— To expand robots capabilities in social and industrial environments, robots will need to adjust to unexpected changes, and interact with a wide variety of objects. One solution being explored is tactile sensing, which can allow a robot to fine-tune its approach to different objects. Current tactile sensors range from simple pressure sensor arrays to information-dense camera arrays. However, the state-of-the-art, data-rich sensors are often large and incur high computational, processing, and power costs. For tactile sensors to be integrated with robots, the sensors would ideally be small, inexpensive, and provide rich, simply-processed data. We present a thin, small tactile sensor that provides pressure, last-mile proximity, and reflectivity using a barometer, short-range time-of-flight sensor, and pulse oximeter. Computational, processing, and power costs remain low. We demonstrate the use of the tactile sensor during object grasping using an anthropomorphic two-finger gripper. Our preliminary results imply the possibility of an algorithm to automate a closed feedback loop between the sensor and a robotic manipulator.**

I. INTRODUCTION

Without tactile sensing, robots are limited in their capability to process their environment. Versatile, easily-manufactured tactile sensors will be crucial to the smooth integration of tactile sensing with existing and new robots in industry and in human-robot interaction.[1],[2]

In industry, robots perform tasks based on control algorithms designed for specific objects. Multi-modal tactile sensors would allow these robots to adaptively distinguish the temperature, texture, shape, and hardness of objects, and use the data in control algorithms for a broader range of objects–thus opening a world of potential for more agile industrial robots.[3],[4] For social and assistive robotics, where environments are unstructured and human-robot interaction is likely to occur, tactile sensing provides data to safely navigate through a chaotic environment, maximizing safe and cautious grasping, movement, and object identification.[1],[2]

Tactile sensors are being developed to meet the needs across this wide range of applications.[1] On the simplest end of the spectrum, pressure sensors can provide the minimum: binary contact information, hardness, and grasping force that can be autonomously controlled.[5] On the other end of the spectrum, cameras are used to gather information-rich data: contact information, shape, texture, hardness, grasping force, slip conditions, and fine pattern recognition.[6] Tactile sensors that provide only contact information and grasping force are limited in their applicability, particularly because contact information is ”on” or ”off” until contact is made with an object.[7] On the other hand, tactile sensors that utilize cameras are bulky, and incur large power, computational and processing costs.[1] These parameters become a nuisance when integrating with robotic manipulators that must maintain dexterity and adaptability.

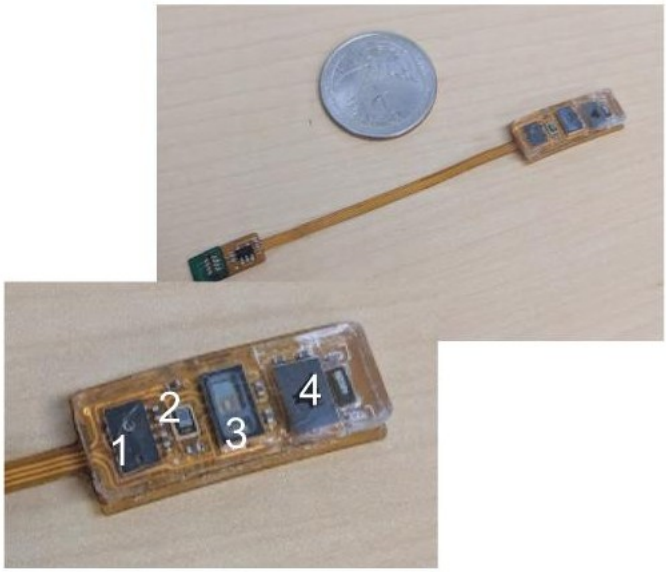


Fig. 1. Modular, multi-modal tactile sensor. Components from left to right: (1) Time-of-flight sensor (2) Pressure sensor (3) Pulse oximeter (4) Accelerometer. A quarter is shown for scale.

We present a multi-modal tactile sensor that is information-rich, small, thin, flexible, and easily integrated with existing models and designs. The sensor employs a barometric pressure sensor, accelerometer, time-of-flight sensor, and pulse oximeter to render contact, temperature, hardness, grasping force, and absolute proximity sensing, with the added capability of taking the vitals signs of humans. We demonstrate the use of this sensor on a two-finger gripper during object manipulation. Our results suggest that the signals from the pressure sensor, time-of-flight sensor, and pulse oximeter could be used as parameters in an autonomous control algorithm. Further work is needed to demonstrate the detection of slip conditions by the accelerometer.

[1]Q. Guengerich is a Chemical Engineering student at the New Mexico Institute of Mining and Technology quintessa.guengerich@student.nmt.edu

[2]E. Markvicka is with the Carnegie Mellon Robotics Institute.

[3]C. Majidi is an Associate Professor of Mechanical Engineering at Carnegie Mellon University.

II. RELATED WORK

The requirements of tactile sensors as well as current pitfalls of various designs are delineated by Kappassov et. al, particularly for those applications in dexterous robot hands. A tactile sensor must measure contact force and surface properties (such as hardness, texture, and shape), and tactile data should be used as a control parameter (force control and shape identification).[1],[5]

Specific design requirements vary by application.[7] Across the board, an emphasis is placed on spatial resolution[1], sensitivity[9], frequency response[6][10], hysteresis[11], wire count[12], and surface friction[1] of the device. Robots tasked with manipulation of small, fragile objects may sacrifice high spatial resolution for higher sensitivity, particularly if wire count must be low. Most other in-hand object manipulation tasks require high spatial resolution.[1] Likewise, tactile sensors that might be used to detect slippage must have a high frequency response, typically sacrificing spatial resolution.[10]

Kappassov et. al also discuss a number of different sensing types, their applications, and their advantages and disadvantages. For example, barometric pressure sensors have been used to control grasping force and distinguish textures to a high resolution,and are listed as having high bandwidth and sensitivity, with low resolution.[9],[8],[1] Comparatively, optical sensors have high resolution, sensitivity, and repeatability, and have been used to determine a vast set of parameters, including shear forces and texture.[6],[13] However optical tactile sensors tend to be relatively large and computationally expensive. Accelerometers and microphones have been used in object detection, grasping, and placement, but signals are prone to noise, which can be worsened by vibrations caused by motors.[1],[14],[10]

With a wide range of sensing modalities available, the balance between meeting specific design requirements while maintaining feasibility can be difficult to strike. Recently, multi-modal tactile sensors strike this balance by addressing these needs in tandem. For instance, the BioTac tactile sensor measures pressure, temperature and vibration to provide high resolution data about texture.[15] Multi-modal tactile sensors are limited by size and wire count in most applications.[1]

In contrast, the multi-modal tactile sensor presented in this work is small and thin. The sensor can be used in a wired configuration, requiring four wires, or in a wireless configuration, sacrificing high frequency and risking packet loss. The sensor detects short-range proximity, pressure, and vibrations, with the added capability of taking the vitals signs of humans via the implementation of a pulse oximeter. In these object grasping experiments, pulse oximeter signals were also observed to represent the reflectivity of each object.

III. METHOD

A. Sensor Fabrication

The tactile sensor consists of the following components:

- Barometer (BMP280, Bosch)
- Absolute orientation sensor consisting of accelerometer, gyroscope, and magnetometer (BNO055, Bosch)
- Time-of-flight sensor (VL6180x, STMicroelectronics)
- Pulse oximeter (MAX30101, Maxim)

The components fit on a flexible board, as pictured in Figure 2. The barometer is prepared using the method described by Tenzer et. al [3], by first casting the barometer in PDMS before removing the cover of the barometer. After component placement, a thin layer of PDMS is added over the board to maintain durability, and provide a uniform point of contact against the sensor.

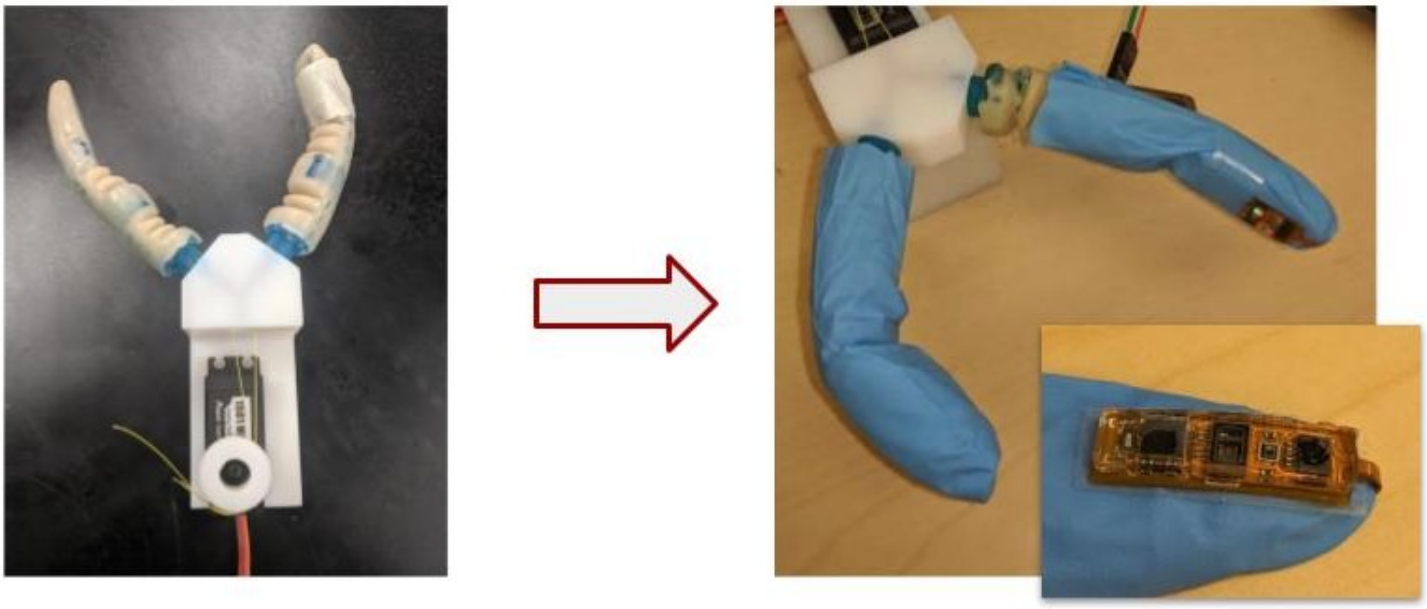


Fig. 2. (Left) Two-finger gripper. (Right) Two-finger gripper with gloves and sensor attached.

B. Construction of a Two-Finger Gripper

A simple two-finger gripper was built, shown in Figure 2, using anthropomorphic fingers inspired by the UC Softhand.[16] Finger endoskeletons were 3D printed from Tough material on Form2 printers, and subsequently cast in Vytaflex 30, a polyurethane elastomer. The fingers were placed onto a 3D-printed mount along with a Power HD 1501 servo, and pulley-actuated using fishing line strung through the inside of the fingers. Nitrile gloves were used to increase the adhesion of the tactile sensors to the fingertip using double-sided tape.

C. Object Manipulation

Object grasping tests were done using an open-loop configuration, collecting data from the sensor while a different program opened and closed the gripper fingers to a predetermined position. These tests were performed in order to demonstrate the most basic signals that could be received from the tactile sensor and inform future work of an autonomously controlled feedback loop. Each object

TABLE I
OBJECT GRASPING SET

Object	Unique property
Black cloth	Soft and non-reflective
Sponge	Soft, elastic, porous
Cardboard	Non-reflective, hard, plyable
Glass slide	Smooth, clear, hard
Shiny coin (quarter)	Small, reflective, hard
Apple	Reflective, soft, firm, large
Blueberry	Dull, small, soft, round
Marshmallow	Soft, elastic
Gummybear	Soft, elastic, translucent, small

was placed between the fingers of the two-finger gripper before data acquisition started. Data was acquired before, during, and after the closing of the gripper. A control test was performed to demonstrate the behavior of the sensors as the fingers closed in on each other, without an object to grasp. Figure 3 below shows the tactile sensor attached to the gripper, in action during object grasping.

IV. RESULTS

Table 2 shows a binary interpretation of the applicability of each signal to an eventual closed-loop algorithm. A data set is marked as not applicable (X) if the test was inconclusive, and marked as applicable (✓) if the data set repeatable and predictable. Time-of-flight proximity data and pulse oximeter reflectivity data is particularly simple to process, with small changes in position simply detected via the implementation of a threshold value. Changes in pressure data would be less detectable. However, three object grasping tasks resulted in steep pressure changes, possibly detectable by a threshold value, and are marked as possible (?).

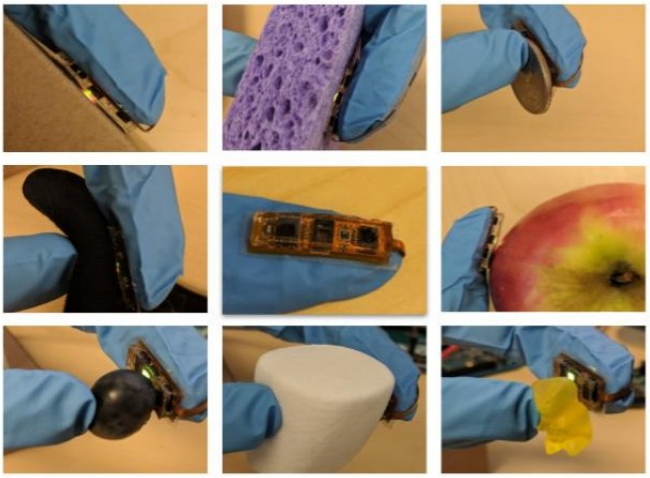


Fig. 3. Grasping experiments were performed to characterize the limits of the tactile sensor. From top left, clockwise: cardboard, sponge, quarter, apple, gummybear, marshmallow, blueberry, black cloth.

TABLE II
APPLICABILITY OF EACH DATA SET TO A CLOSED LOOP ALGORITHM

Object	Pressure	ToF	Pulse Ox.
Black cloth	X	✓	✓
Sponge	X	✓	✓
Cardboard	X	✓	✓
Glass slide	X	✓	✓
Shiny coin (quarter)	?	✓	✓
Apple	X	✓	✓
Blueberry	?	✓	✓
Marshmallow	?	✓	✓
Gummybear	X	✓	✓

A. Pressure Sensor Data

Pressure sensor data indicated that for each object, pressure decreased upon contact with the object. This response is the inverse of what we expected to find: an increase in pressure as contact is made and force increases. Furthermore, the baseline pressure reading for the control read a number much higher than atmospheric pressure. We conclude that the baseline pressure of this sensor is modified by strain on the PDMS elastomer, residual from the curing process.

However, the pressure data was not inconclusive. The largest pressure differentials occurred during the manipulation of the marshmallow and the blueberry, and the smallest pressure differential occurred during manipulation of a piece of cardboard. We conclude that flat objects distribute a load across the face of the tactile sensor, while rounded faces load pressure onto a tangent point. One exception to this observation occurs during the manipulation of a coin, where we expected and observed a high pressure differential because the coin is non-deformable.

B. Time-of-Flight Sensor Data

Time-of-flight sensor data indicated out-of-range data ($>$200 mm) until the sensor came across the face of the object. At that point, range data decreased to it's baseline

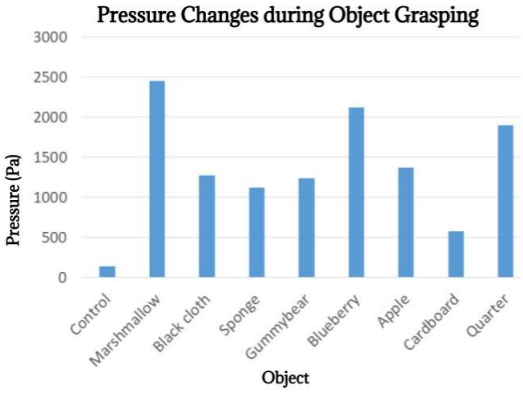


Fig. 4. Pressure differential during each object manipulation experiment. Units in Pascals.

value, flatlined while the gripper was closed, increased, and indicated out-of-range data again. This is the predicted outcome of the sensor when used in a configuration other than parallel-plate gripping, during which the sensor would always be parallel to a surface (thus never indicating out-of-range data). Although out-of-range data points would have to be accounted for, a simple closed-loop algorithm to guide the gripper closed could easily be structured based on proximity data.

C. Pulse Oximetry Data

Pulse oximetry data exhibited an inverse signal to the time-of-flight signal: as the sensor approached each object, reflection of IR light and red light increased until the object was gripped, at which point the signal stayed the same until the object was released again. Black cloth did not reflect any red light, but the sponge and marshmallow reflected large amounts of red light. There was a detectable change in IR light for every object, easily detectable by a simple heuristic closed-loop algorithm.

V. CONCLUSIONS

This preliminary data implies that a closed loop algorithm will be possible to implement on this sensor and gripper together. Time-of-flight proximity data indicated a repeatable, predictable change that can be interpreted by a closed-loop algorithm to control approach to an object. Upon contact, pressure data can be used to control grasping.

However, pressure experiments will need to be repeated with a pressure sensor that's baseline exists at atmospheric pressure, to confirm the magnitude and behavior of the pressure sensor upon contact with different objects.

Future work includes building a new sensor and repeating these tests, as well as analyzing results from slip condition detection testing. We would also like to repeat these tests on a dexterous 5-finger robotic hand. Finally, the implementation of a closed-loop algorithm will be crucial in the characterization of this sensor.

ACKNOWLEDGMENT

This research was supported by the Carnegie Mellon Robotics Institute, the National Science Foundation, and the Office of Naval Research.

Quintessa would like to thank Dr. Eric Markvicka for his mentorship and Dr. Carmel Majidi for his advising and leadership.

REFERENCES

[1] Z. Kappassov, J. Corrales and V. Perdereau, "Tactile sensing in dexterous robot hands Review", Robotics and Autonomous Systems, vol. 74, pp. 195-220, 2015.

[2] Stillfried G., Hillenbrand U., Settles M., van der Smagt P. (2014) MRI-Based Skeletal Hand Movement Model. In: Balasubramanian R., Santos V. (eds) The Human Hand as an Inspiration for Robot Hand Development. Springer Tracts in Advanced Robotics, vol 95. Springer, Cham

[3] euRobotics aisbl, Robotics 2020. strategic research agenda for robotics in Europe. $https : //www.eu - robotics.net/cms/upload/topic_groups/SRA2020_SPARC.pdf$ (accessed 07.08.2018).

[4] M. Lee and H. Nicholls, "Review Article Tactile sensing for mechatronicsa state of the art survey", Mechatronics, vol. 9, no. 1, pp. 1-31, 1999.

[5] Dahiya R.S., Valle M. (2013) Tactile Sensing Technologies. In: Robotic Tactile Sensing. Springer, Dordrecht

[6] Yussof H., Ohka M., Suzuki H., Morisawa N. (2009) Tactile Sensing-based Control System for Dexterous Robot Manipulation. In: Ao SI., Rieger B., Chen SS. (eds) Advances in Computational Algorithms and Data Analysis. Lecture Notes in Electrical Engineering, vol 14. Springer, Dordrecht

[7] L. Harmon, "Automated Tactile Sensing", The International Journal of Robotics Research, vol. 1, no. 2, pp. 3-32, 1982.

[8] Y. Tenzer, L. Jentoft and R. Howe, "The Feel of MEMS Barometers: Inexpensive and Easily Customized Tactile Array Sensors", IEEE Robotics & Automation Magazine, vol. 21, no. 3, pp. 89-95, 2014.

[9] J. A. Corrales Ramn, V. Perdereau and F. Torres Medina, "Multi-fingered robotic hand planner for object reconfiguration through a rolling contact evolution model," 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, 2013, pp. 625-630.

[10] M. T. Francomano, D. Accoto and E. Guglielmelli, "Artificial Sense of SlipA Review," in IEEE Sensors Journal, vol. 13, no. 7, pp. 2489-2498, July 2013.

[11] L. Odhner, L. Jentoft, M. Claffee, N. Corson, Y. Tenzer, R. Ma, M. Buehler, R. Kohout, R. Howe and A. Dollar, "A compliant, underactuated hand for robust manipulation", The International Journal of Robotics Research, vol. 33, no. 5, pp. 736-752, 2014.

[12] R. S. Dahiya, P. Mittendorfer, M. Valle, G. Cheng and V. J. Lumelsky, "Directions Toward Effective Utilization of Tactile Skin: A Review," in IEEE Sensors Journal, vol. 13, no. 11, pp. 4121-4138, Nov. 2013.

[13] R. Li, E.H. Adelson (2013), "Sensing and Recognizing Surface Textures Using a GelSight Sensor", IEEE Conference on Computer Vision and Pattern Recognition. 2013

[14] J.M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, K.J. Kuchenbecker "Human-Inspired Robotic Grasp Control With Tactile Sensing" IEEE Transactions on Robotics, vol. 27, 2011

[15] Wettels N., Fishel J.A., Loeb G.E. (2014) Multimodal Tactile Sensor. In: Balasubramanian R., Santos V. (eds) The Human Hand as an Inspiration for Robot Hand Development. Springer Tracts in Advanced Robotics, vol 95. Springer, Cham

[16] M. Tavakoli, R. Batista and L. Sgrigna, "The UC Softhand: Light Weight Adaptive Bionic Hand with a Compact Twisted String Actuation System", Actuators, vol. 5, no. 1, p. 1, 2015.

Trajectory Prediction of a Fixed-Wing UAV with Sequence-to-Sequence RNNs

Adarsh Karnati[1] and Sebastian Scherer[2]

***Abstract*—The growing use of Unmanned Aerial Vehicles (UAVs) has prompted extensive research into risk-aware motion planning. In particular, UAVs require onboard contingency planners, which safely land the robot in emergency situations. To make intelligent decisions, contingency planners must be aware of the UAVs current dynamics, which may change due to weather or propulsion failures. Current methods rely on fixed models, which can be simplistic or make strong assumptions, and can fail to capture these changes in dynamics. We address this issue by measuring risk through a sequence-to-sequence recurrent neural network (RNN). The model is made of an encoder RNN, which takes in a sequence of observations in real time to build an internal state of the UAVs modified dynamics. Then, a decoder RNN uses the encoder's internal state and a proposal path to produce a distribution of states relative to this proposed path. We compare the sequence-to-sequence model with a Sequential Monte Carlo method in scenarios with varying wind conditions for a simulated fixed-wing UAV.**

***Index Terms*—unmanned aerial vehicle, contingency planner, recurrent neural network, sequence-to-sequence, encoder, decoder**

I. Introduction

Since the birth of modern robotics, the potential of autonomous vehicles for use in every-day life has moved closer to reality. Unmanned Aerial Vehicles (UAVs), for example, have made forays in agriculture, search-and-rescue and delivery scenarios [8]. Fixed-wing UAVs, specifically, present enormous opportunity in that they can reach higher speeds and fly more efficiently than their rotorcraft counterparts [7].

However, the limiting factor in the deployment of aerial drones is the ability to ensure safe operation around people and infrastructure. The cornerstone of UAV safety is contingency planning [9] intelligent decision making in the event of a necessary landing. The primary role of a contingency planner is to quickly find a feasible (obstacle-free) path to a low risk terminal state [16]. To this end, contingency planning is concerned with measuring risk and producing safety guarantees for proposed paths.

Risk in robotics motion planning requires an understanding of the robot workspace as well as transition uncertainty [7]. Probabilistic modeling of risk often considers the tracking distribution, which is the distribution over actual trajectories the robot could realize while attempting to follow a proposed reference path. Using functions of the tracking distribution of a candidate path to optimize on [3] or rank paths allows a planner to iteratively find low risk paths.

NASA UAS Contingency Grant
[1]University of California, Berkeley: akarnati@berkeley.edu
[2]Robotics Institute, Carnegie Mellon University: basti@andrew.cmu.edu

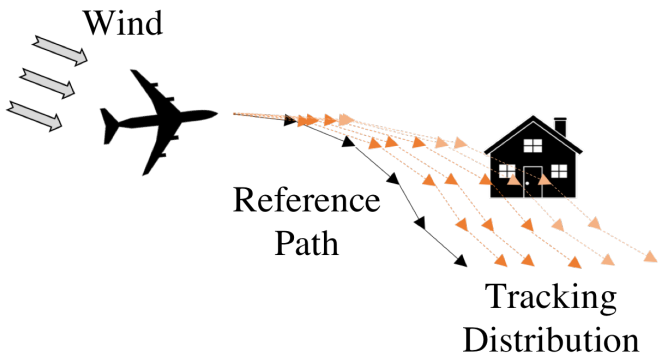


Fig. 1. Example of a tracking distribution of a fixed wing UAV coinciding with a high risk obstacle (house)

However, representing and operating on the tracking distribution of a robot of several dimensions presents high computational overhead. Moreover, tracking distributions are dependent on several factors such as a reference path, world conditions and maneuver capabilities of the robot. Some of these conditions are unknown, partially known or can only be observed indirectly in real-time.

These constraints generally lead roboticists to approximate the system and tracking distribution with parametric (model-based) techniques, in which an underlying probabilistic model is assumed and reasoned on [1]. One of the most celebrated classes of parametric estimators is the Sequential Monte Carlo (SMC) family. In SMC approaches, particles representing possible robot states are propagated through the underlying dynamics model. However, these models often are too simplistic or make strong assumptions, producing poor predictions. Also, power failures aboard the robot, may change the dynamics model completely, e.g. the dynamics of an engine-enabled fixed-wing UAV is far different from that of a gliding one. Model-based algorithms may not be able to rectify these changes through simple parameter estimation.

This study tackles these issues by proposing a non-parametric, data-driven approach to the trajectory distribution prediction problem for use in a contingency planner. The method relies on a recurrent neural network (RNN), sequence-to-sequence architecture. RNNs are feedforward neural networks that accommodate inputs of variable length or temporal dependencies. Sequence-to-sequence models make use of two RNNs, an encoder, which takes a sequence of inputs, and a decoder, which uses the internal state of the encoder RNN

to produce an output sequence. At a high level, the authors approach is to use the encoder RNN to build an implicit model of the worlds dynamics by inputting a sequence of observations. Then the decoder uses this model to produce the tracking distribution relative to a candidate path.

The main contributions of this paper are summarized below:

- Formulation of risk assessment in terms of the tracking distribution
- A sequence-to-sequence architecture for predicting the tracking distribution from real time observations
- Performance comparison of the sequence-to-sequence model with a common SMC approach on a simulated dataset of a fixed wing UAV

II. Related Work

Our work is primarily related to contingency path planning and subsequent tracking in uncertain environments. The following section is split into two parts. First, we discuss primary facets of contingency planning, specifically optimality, computation speed and safety guarantees. Then, we describe risk analysis and its role as the foundation of contingency planners.

A. Contingency Planning

Optimal path finding is producing the shortest path to a goal based on some cost function. It is most commonly solved with search-based methods over a graph of robot states [13]. However, basic implementations of A*, D* and other shortest-path algorithms suffer from large computation overhead to be effective in high-dimension state and action spaces [12]. Dubins paths [11] have been used extensively for fast and optimal path planning of non-holonomic systems. However, Dubins planners cannot consider obstacles and may not produce trackable paths in uncertain systems [12].

Sampling based planners have become popular in recent years for their ability to quickly find feasible paths in environments with obstacles [13]. However, the speed of the algorithm largely depends on heuristics or scoring functions similar to the search methods above [12]. Some variants, such as RRT*, and RRG*, also provide asymptotic optimality guarantees [6]. Although these algorithms have been used successfully in many motion planning scenarios, they may not provide safely trackable paths for systems with complicated dynamics and uncertainty [5].

Online trajectory optimizers, such as CHOMP, TrajOpt and STOMP, have also been developed recently, and attempt to minimize an objective function over candidate paths [3], [10].These candidate paths are often produced using the aforementioned sampling methods and allow for explicit incorporation of risk, via the objective function [3]. This flexibility also means that design of the objective function is often handmade, which can incorporate bias and fail to capture subtleties of the system.

B. Risk Modeling

As mentioned in the previous section, contingency planning is based on an underlying score function. This function is a representation of risk and has been defined several ways in literature. The most basic of risk modeling is the stopping distance between a robot and a sensed obstacle [2]. Stopping distance is an example of a conservative model that acts on worst case scenarios. Robots that use conservative models do not exploit the full capabilities of their dynamics, which can cause greater danger in time-critical situations. Another example of a conservative risk modeling is reachability analysis, which is the prediction of whether a robot will collide with an obstacle. Previously a computation process suffering from the curse of dimensionality, reachability was made possible to run online in [4]. However, reachability analysis is fundamentally a binary measure of risk and is not the most informative descriptor.

Other risk models reason on expected scenarios, rather than worst case analysis. [2] features a selection of emergency maneuvers that guarantee safety in expected collision scenarios. However, this library was constructed for a quadrotor UAV, and its maneuvers consist of hovering patterns, which cannot be executed by a fixed-wing aircraft. Additionally, [13] uses a Long Short-Term Memory (LSTM) RNN to quickly estimate a low-risk path for a quadcopter. This model-free approach relies on prior knowledge of the environments dynamics, which is a strong assumption to make in an emergency scenario.

III. Problem Formulation

A. State Space Model

The state space model of the fixed-wing UAV is described by table 1 and equation 1, [14]:

Table 1: State space variables

Name	Description
x	Inertial north position of UAV (north-east-down)
y	Inertial east position of UAV (north-east-down)
z	Inertial down position of UAV (north-east-down)
u	Body frame velocity along roll axis
v	Body frame velocity along pitch axis
w	Body frame velocity along yaw axis
ϕ	Roll
θ	Pitch
ψ	Yaw
p	Roll rate
q	Pitch rate
r	Yaw rate
f_x, f_y, f_z	External forces along roll, pitch and yaw axes
$\boldsymbol{J}$	UAV inertia matrix
$\boldsymbol{m^b}$	External moments about roll, pitch and yaw axes
$\boldsymbol{w_c}$	Constant wind vector in world frame
$\boldsymbol{w_g}$	Turbulence vector in body frame

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi\\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi\\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta\end{bmatrix}\begin{bmatrix}u\\ v\\ w\end{bmatrix} \tag{1}$$

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \begin{bmatrix}rv - qw\\ pw - ru\\ qu - pv\end{bmatrix} + \frac{1}{m}\begin{bmatrix}f_x\\ f_y\\ f_z\end{bmatrix} \tag{2}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & s_\phi t_\theta & c_\phi t_\theta\\ 0 & c_\phi & -s_\phi\\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \boldsymbol{J^{-1}}\left[-\begin{bmatrix}p\\ q\\ r\end{bmatrix} \times (\boldsymbol{J}\begin{bmatrix}p\\ q\\ r\end{bmatrix}) + \boldsymbol{m^b}\right] \tag{4}$$

We model atmospheric disturbances as the sum of **constant wind** and **turbulence**. Both variables fully define the vehicle airspeed, angle of attack, and side-slip angle used to derive the external forces (see [14] pg. 59).

$$\begin{aligned}\boldsymbol{f^b} =& f(\boldsymbol{w_c}, \boldsymbol{w_g})\\ \boldsymbol{w_c} \sim& \mathcal{N}(\mu_w, \Sigma_w)\\ w_g^i(s) =& \sigma_u\sqrt{\frac{2V_a}{L_u}}\frac{1}{s + \frac{V_a}{L_u}}\\ w_g^j(s) =& \sigma_v\sqrt{\frac{3V_a}{L_v}}\frac{s + \frac{V_a}{\sqrt{3}L_v}}{(s + \frac{V_a}{L_u})^2}\\ w_g^k(s) =& \sigma_w\sqrt{\frac{3V_a}{L_w}}\frac{s + \frac{V_a}{\sqrt{3}L_w}}{(s + \frac{V_a}{L_w})^2}\end{aligned} \tag{5}$$

$\boldsymbol{w_g}$ is modeled using the Dryden gust filter [14], which maps white noise to body velocities. Where $\sigma_u, \sigma_v, \sigma_w$ and L_u, L_v, L_w are intensity hyperparameters and spatial wavelengths, respectively, along the vehicle frame axes. V_a is vehicle airspeed.

Our linear observation model is defined in equation 6.

$$\begin{aligned}\boldsymbol{y} =& B\boldsymbol{x} + \boldsymbol{v}\\ \boldsymbol{v} \sim& \mathcal{N}(0, \Sigma_v)\end{aligned} \tag{6}$$

B. Trajectory Nomenclature

We define a **reference trajectory** $\zeta(t)_{ref}$ as a sequence of N evenly spaced robot states $(x_1, ..., x_N)$, each of dimension d for some duration T.

$$\zeta_{ref}(t) := (x_1, ..., x_N) \tag{7}$$

Let $\zeta_{act}^{(i)}(t)$ be an **actual trajectory**; a random realization of a $\zeta_{act}(t)$ from the true tracking distribution, p_{true}, given a world, $\mathcal{W}$, and control inputs, $\mathcal{U}$, of the robot. $\mathcal{W}$ is used to represent world uncertainty. A concrete example for $\mathcal{U}$ could be the roll, pitch, yaw and thrust on a fixed-wing UAV.

$$\begin{aligned}\zeta_{act}(t) :=& (X_1, ..., X_N) \sim p_{true}(\cdot|\mathcal{W}, \mathcal{U})\\ \zeta_{act}^{(i)}(t) :=& (X_1 = x_1, ..., X_N = x_n)\end{aligned} \tag{8}$$

A **model-based** representation of $\mathcal{W}$ and $\mathcal{U}$, assumes that ζ_{act} can be fully represented by a first-order differential equation with parameters derived from the world and and control sets.

$$\begin{aligned}\dot{x} =& f(x, \boldsymbol{u}, \boldsymbol{w})\\ u \sim& \mathcal{U}\\ w \sim& \mathcal{W}\end{aligned} \tag{9}$$

A **model-free** approach, however, relies exclusively on observations to represent the environment:

$$\zeta_{obs}(t) := (Y_1, ..., Y_k) \tag{10}$$

IV. Approach

A. Distribution Representation

In this study, $p_{true}(\cdot|\mathcal{W}, \mathcal{U})$ must be approximated via simulation in order to be used as **training labels**. First, for computation reasons, we describe the full trajectory distribution as a vector of N conditional distributions, which we denote below, dropping the conditionals for clarity:

$$p_{true}(\cdot|\mathcal{W}, \mathcal{U}) := \big[p_1(\cdot), ..., p_N(\cdot)\big] \tag{11}$$

Next, each distribution is discretized to a k-dimension histogram, k $\leq$ d, of numbered voxels α. Visually, the histogram is centered on the reference state at each timestep, which reduces the required size of the grid. This histogram allows us to approximate the true distribution via a normalized sum of the occupied voxels over a sample of $M, M >> 1$, actual trajectories.

$$p_\tau(\alpha_j) \approx \frac{1}{M}\sum_i^M \mathbb{I}\{\int_{\alpha_j} \zeta_{act}^{(i)}(\tau)dx > 0\} := P_\tau(\alpha_j) \tag{12}$$

The above formula is simply an unbiased estimator for $\mathbb{E}[g(\zeta_{act}^{(i)}(\tau))]$ where $g(\cdot)$ is the indicator function. In other words, we are trying to estimate the expectation of every voxel of the discretized distribution, which is guaranteed to converge almost surely as $M \to \infty$, by the Strong Law of Large Numbers.

Now we define the **prediction of the sequence-to-sequence** model as $\hat{p_t}(\cdot|\zeta_{ref}(t), \zeta_{obs}(t), \theta)$, the neural net weights parameterized by θ. The prediction can then be defined as an optimization problem, which can be solved through gradient methods.

$$\hat{\theta} = argmin\sum_i^N D_{KL}(P_i||\hat{p_i}) \tag{13}$$

Where $D_{KL}(p||q)$ is the Kullback-Leibler Divergence, a common premetric used to relate the information difference

of two distributions. The $D_{KL}(p||q)$ will also be used as an accuracy measure.

$$D_{KL}(p||q) := \sum_x p(x) log(\frac{p(x)}{q(x)}) \tag{14}$$

B. Data Generation

The model simulator plans paths using a x-y-z-heading Dubins path planner, then filters this path by simulating tracking the path, in perfect conditions, with a Pure Pursuit follower and PID loop closure control [14]. This dynamically filtered path serves as the reference trajectory defined above. The simulation is then run with the same Pure Pursuit follower tracking the reference trajectory, but now incorporating atmospheric disturbances. This wind is modeled as described in the Problem Formulation section. Specifically, w_c is drawn at the beginning of a simulation, and remains constant in speed and direction for the entirety of the simulation. Whereas the w_g is drawn every timestep of the simulation.

The dataset consists of ∼4000 reference Dubins trajectories, with terminal x-y-z-θ states in randomly chosen in $[-150, 150]^2 \times [-25, 25]$. This region of goal states was chosen based on the simulated UAV's physical parameters and to keep trajectories relatively short (∼60 seconds). In planning the reference trajectories, a constant target speed of 20 m/s is used. The reference trajectories are randomly and evenly split into four groups, each associated with weather conditions: low-wind-low-turbulence (LWLT), low-wind-high-turbulence (LWHT), high-wind-low-turbulence (HWLT) and high-wind-high-turbulence (HWHT). The weather conditions are used in running the simulations of trajectory tracking, and the parameters of these conditions are summarized in Table 2.

For each reference trajectory, 200 simulations are run at a timestep resolution of 0.01 seconds and downsampled with an interval of 1 second. These simulations were used to build histogram approximations of the true tracking distribution, with a bin size of 2 meters^3 and range of (-30,30)^3 meters. The simulations were run on 4 Pascal Titan X processors in parallel.

Table 2: Weather condition parameters

	$\boldsymbol{w_c}$	$\boldsymbol{w_g}$
LWLT	$\lvert w_c\rvert \sim \mathcal{N}(1,1)\frac{m}{s}$ $\angle w_c \sim$ Uni$[0, 2\pi)$	$L_u, L_v = 200m, L_w = 50m$ $\sigma_u, \sigma_v = 1.06\frac{m}{s}, \sigma_w = 0.7\frac{m}{s}$
LWHT	$\lvert w_c\rvert \sim \mathcal{N}(1,1)\frac{m}{s}$ $\angle w_c \sim$ Uni$[0, 2\pi)$	$L_u, L_v = 200m, L_w = 50m$ $\sigma_u, \sigma_v = 2.12\frac{m}{s}, \sigma_w = 1.4\frac{m}{s}$
HWLT	$\lvert w_c\rvert \sim \mathcal{N}(2,1)\frac{m}{s}$ $\angle w_c \sim$ Uni$[0, 2\pi)$	$L_u, L_v = 200m, L_w = 50m$ $\sigma_u, \sigma_v = 1.06\frac{m}{s}, \sigma_w = 0.7\frac{m}{s}$
HWHT	$\lvert w_c\rvert \sim \mathcal{N}(2,1)\frac{m}{s}$ $\angle w_c \sim$ Uni$[0, 2\pi)$	$L_u, L_v = 200m, L_w = 50m$ $\sigma_u, \sigma_v = 2.12\frac{m}{s}, \sigma_w = 1.4\frac{m}{s}$

C. Sequence-to-Sequence Model

The architecture for the model can be seen in Figure 2. The model was trained end-to-end using back-propagation in time. Provided a sequence of observations, the model produced a sequence output scored with the KL-Divergence premetric.The RNN was not provided any information regarding the weather conditions of the trajectory it was predicting, other than the observations. This network was trained on roughly 75% of the dataset and tested on the remaining 25%.

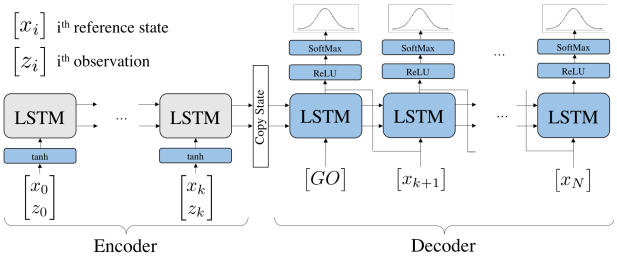


Fig. 2. Sequence-to-sequence architecture used in the experiments

D. Unscented Kalman Filter Baseline

The sequence-to-sequence model was compared to an Unscented Kalman Filter (UKF) on the same test set of trajectories. The UKF uses the Expectation-Maximization algorithm to estimate the variance matrix of additive, zero-mean process noise, which is assumed as the model of environmental disturbances.

V. Results

Figures 3 shows the predicted distributions provided by the UKF after performing parameter estimation with a previous trajectory of observations. Figures 3 also shows the predicted distributions provided by the sequence-to-sequence model with the same number of observations. Qualitatively, it is clear that the variance of the UKF grows quite rapidly after the progression of a turn. Additionally, the UKF's predicted distributions are Gaussian in distribution and biased towards the reference path rather than the mean of the actual trajectories.

In contrast, the predicted distributions of the sequence-to-sequence model are much more tightly clustered around the true tracking distribution. The shapes of the sequence-to-sequence predictions are also noticeably more similar to the true distributions', suggesting a less biased estimate.

Quantitatively, the KL-Divergence evaluation of the UKF vs. the sequence-to-sequence model is summarized in Table 3. We see that our RNN model produces a distribution that is around 6 bits closer to the true tracking distribution than the UKF, for all weather conditions. These results suggest that the model is able to differentiated between weather conditions and use this information to provide better predictions. Moreover, the sequence-to-sequence model outperforms the UKF without an explicit definition of a dynamics model.

Table 3: D_{KL} Between Predictions and Ground Truth (bits)		
	UKF	Sequence-to-Sequence
LWLT	21.260	14.600
LWHT	22.437	16.714
HWLT	24.950	17.390
HWHT	25.029	19.560

VI. Conclusion and Future Work

In this work we presented a prediction method for quantifying risk in emergency landing scenarios. We formulated an understanding of risk via the tracking distribution, a construct that many model-based prediction methods fail to capture. Our model-free proposal, the sequence-to-sequence RNN, outperformed a model-based counterpart in predicting the tracking distribution of four different weather conditions.

For the future, the model can be improved by changing the underlying representation of the true tracking distribution. Instead of a sparse histogram that is currently being used, a set of polynomials may provide a richer and lower dimension approximation. The sequence-to-sequence networks should also be compared to more powerful density estimators such as a Gaussian Process filter. Additionally, evaluating the baseline and our model with more physically intuitive metrics, such as cross-track error may be beneficial. Finally, extending the dataset to include a wider diversity of weather conditions as well as propulsion failures would provide more insight into our model's capabilities.

VII. Acknowledgements

The authors would like to thank Rachel Burcin, John Dolan and Ziqi Guo for their tireless efforts in coordinating the Robotics Institute Summer Scholars program and for funding the authors' research.

References

[1] V. Akbarzadeh, C. Gagn and M. Parizeau,*"Kernel density estimation for target trajectory prediction," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 3449-3456.*

[2] S. Arora, S. Choudhury, D. Althoff and S. Scherer,*"Emergency maneuver library - ensuring safe navigation in partially known environments," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 6431-6438.*

[3] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor and S. Schaal, *"STOMP: Stochastic trajectory optimization for motion planning," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 4569-4574.*

[4] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac and C. J. Tomlin, *"FaSTrack: A modular framework for fast and guaranteed safe motion planning," 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, 2017, pp. 1517-1522.*

[5] Richter C., Bry A., Roy N. *(2016) Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In: Inaba M., Corke P. (eds) Robotics Research. Springer Tracts in Advanced Robotics, vol 114. Springer, Cham*

[6] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, textit "Anytime Motion Planning using the RRT*," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 1478-1483.

[7] L. N. Mascarello and F. Quagliotti, Analysis of Safety Requirements for Small Unmanned Aerial Systems (sUAS), Handbook of Unmanned Aerial Vehicles, pp. 129, 2018.

[8] Puri, A. A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance; Department of Computer Science and Engineering, University of South Florida: Tampa, FL, USA, 2005; pp. 129.

[9] J. DiFelici and C. Wargo, "UAS safety planning and contingency assessment and advisory reseach," 2016 Integrated Communications Navigation and Surveillance (ICNS), Herndon, VA, 2016, pp. 8E3-1-8E3-16.

[10] John Schulman , Yan Duan , Jonathan Ho , Alex Lee , Ibrahim Awwal , Henry Bradlow , Jia Pan, Sachin Patil , Ken Goldberg , Pieter Abbeel, Motion planning with sequential convex optimization and convex collision checking, International Journal of Robotics Research, v.33 n.9, p.1251-1270, August 2014.

[11] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. Amer. J. Math., 79:497516, 1957.

[12] R. Pepy, A. Lambert and H. Mounier, "Path Planning using a Dynamic Vehicle Model," 2006 2nd International Conference on Information Communication Technologies, Damascus, 2006, pp. 781-786.

[13] H. Brahmi, B. Ammar and A. M. Alimi, "Intelligent path planning algorithm for autonomous robot based on recurrent neural networks," 2013 International Conference on Advanced Logistics and Transport, Sousse, 2013, pp. 199-204.

[14] Randal W. Beard , Timothy W. McLain, Small Unmanned Aircraft: Theory and Practice, Princeton University Press, Princeton, NJ, 2012.

[15] B. Luders "Robust trajectory planning for unmanned aerial vehicles in uncertain environments" 2008.

[16] D. Althoff, M. Althoff and S. Scherer, "Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 3470-3477.

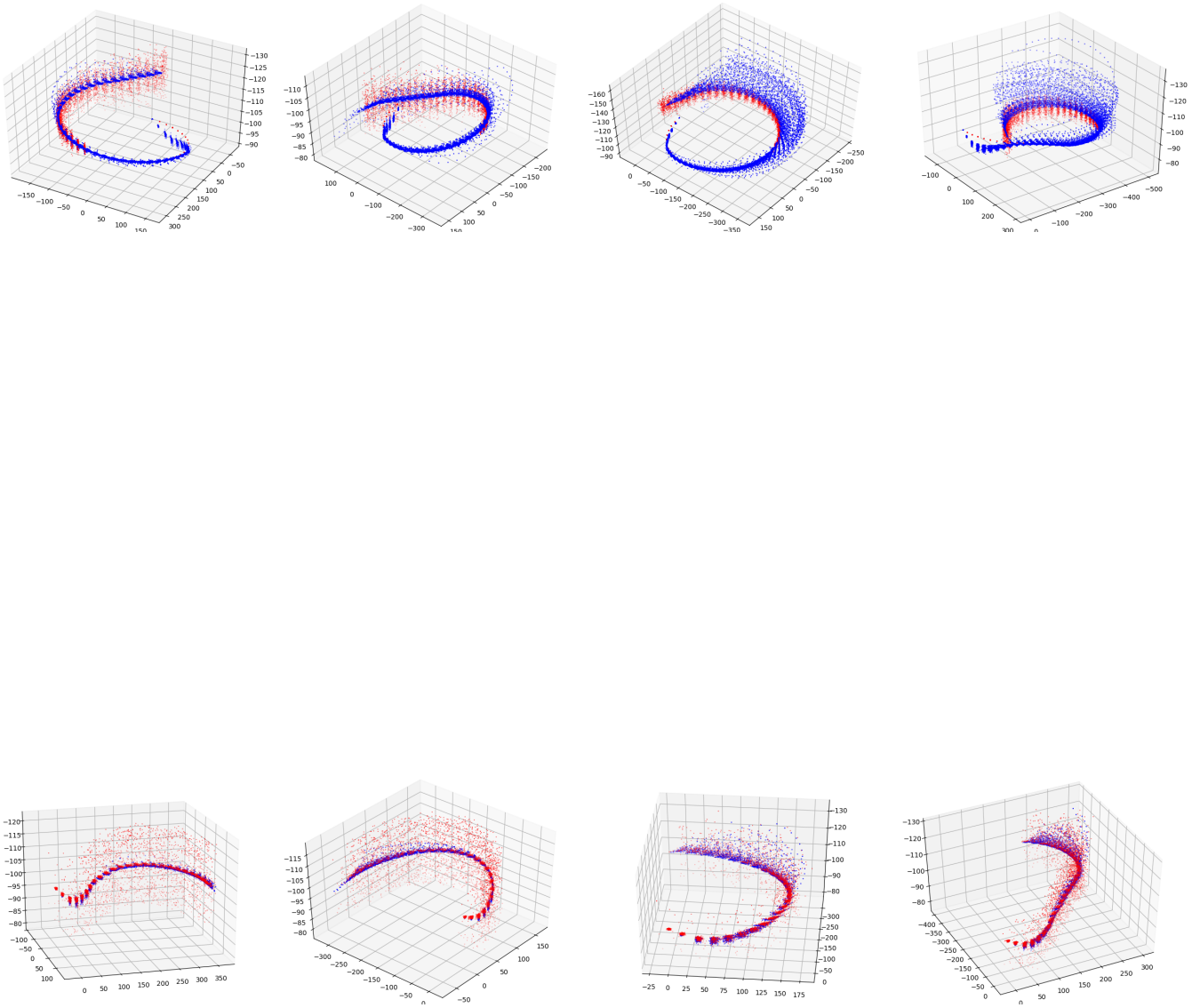


Fig. 3. (Top) predicted distributions of UKF in red vs ground truth in blue. (Bottom) predicted distributions of sequence-to-sequence in red vs ground truth in blue

Multi-UAV Persistent Coverage for Multiple Target Visitation Frequencies

Tushar Kusnur[1,*]

***Abstract*— In this paper, we focus on the problem of persistent monitoring, cast as a coverage problem for multiple unmanned aerial vehicles in a discrete environment. We also account for kinodynamic constraints and resource constraints on the UAVs. With multiple coverage zones of various priorities, there are multiple objectives to be optimized for viz. area covered, travel time, travel distance, etc. Our goal is to develop a lattice-based plan that allows us to compute bounded suboptimal results in a reasonable time with re-planning and partial-plan improvement capabilities. We would also like to prioritize over the multiple objectives to optimize. We will demonstrate our method with the help of a custom visualization tool developed to replicate this scenario specifically as accurately as possible.**

I. INTRODUCTION

Long-term and persistent surveillance and monitoring tasks can be too risky or repetitive for humans to perform. Thus, it is common for robots to be deployed for these purposes. In scenarios that entail surveillance or monitoring in a disaster zone or military operation zone, multi-robot teams can cooperate to speed up the process of setting up an infrastructure and deploying a fleet of agents to perform targeted sensing for the entire environment. If the area to be monitored is significantly large, a small number of robots cannot efficiently cover all the desired locations simultaneously. Also, in such a scenario, one might want to go back to certain regions multiple times to check for any developments or threats. We define our problem as one of persistent coverage for a fleet of agents to constantly revisit targeted locations in an environment with multiple, pre-defined visitation frequencies. Additionally, we must incorporate possibilities of drained resources and consequent departure of agents from the scene for recharging purposes, kinodynamic constraints, and significantly large environments in our framework. We demonstrate our method on a simulation via a custom visualization framework and deploy it on Unmanned Aerial Vehicles (UAVs), although the basic algorithmic concepts can be applied to more platforms.

Numerous tasks like surveillance, mapping, and mail delivery require visiting a series of targets or points or traversing parts of a large environment to accomplish a goal. Many times this goal is to efficiently cover the entire area by visiting all such points or parts of the environment. This task is also referred to as Coverage Path Planning (CPP). The survey on CPP for Robotics, [1], by Galceran and Carreras covers most of the traditional methods of solving the general CPP. This task is integral to many robotic applications, such as vacuum cleaning robots [2], painter robots [3], lawn mowers [4], and inspection of complex underwater structures [5], just to name a few. Early methods for solving the Coverage Path Planning problem use classical cell decomposition (exact and approximate) and also coverage approaches based on detection of landmarks. [6] presents coverage algorithms for environments that can be represented as graphs for single and multiple robots in a system.

Given the environment beforehand, with information about the targeted regions and their visitation frequencies, we can approximate travel times between them via information about vehicles' sizes, velocities, and some initial pre-planning. Having done this, it is natural to think of this coverage problem as a form of the graph problem of optimal node visitation. This leads us to the classical Traveling Salesman Problem (TSP). Though the TSP solution is an optimal patrol for one agent, it can be used as a basis for solving the problem for multiple agents by either distributing the agents evenly among the solution, or by dividing the graph into subgraphs and solving multiple simultaneous TSPs.

[7] use a variant of a frontier-based approach with multiple objective utility functions for the problem of exploring an environment. They incorporate soft constraints on maximum and minimum inter-robot distances and a heading bias.

[8], [9] address the Multi-Robot Persistent Coverage Problem (MRPCP) as a variant of the Vehicle Routing Problem (VRP). For VRP and its variants, one must find the optimal routing strategy that allows a fleet of vehicles to visit a set of targets while trying to minimize some objective, which usually takes the form of travel distance. They consider an objective function that minimizes the maximum period of tours, wherein the solutions include a path that visits all targets once and then revisits the first target. The key distinction of MRPCP from traditional VRP is that the robots have a limited fuel capacity and operate on a time scale much longer than the capacity allows. The VRP at its most basic is NP-hard. Therefore, solutions in practice rely on heuristic methods to provide suboptimal routes or formulate the problem as a Mixed Integer Program (MIP) and solve to optimality using standard Branch-and-Bound techniques.

In our work, we attempt to modify the frontier-based exploration method to a frontier-based coverage method and defining our objectives based on the visitation frequencies of the targets in the environment, as well as other quantities like traversal cost, energy/fuel consumption, and exit of one of the UAVs in the middle of a plan for recharging/refueling.

*The author was supported by the Federation of Indian Chambers of Commerce and Industry (FICCI).

[1]Tushar Kusnur is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A. kusnur tushar at gmail dot com

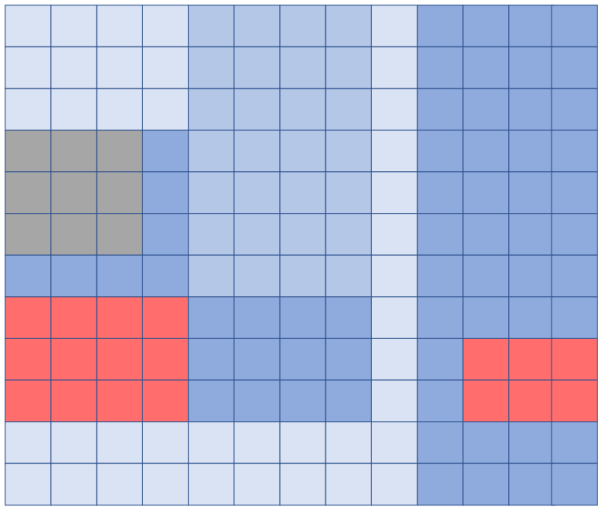
We demonstrate our method via visualization and debugging on a custom-made Graphical User Interface using the Qt framework and also by comparing it with existing algorithms that are applicable to our problem.

II. THE PROBLEM

Consider an $m \times n$ grid representing a discrete 2D environment that needs to be efficiently monitored by N UAVs. Each cell of the grid contains a number determining the 'revisit time' of that cell. A revisit time $r_{i,j}$ implies that cell (i, j) should be revisited by a UAV after $r_{i,j}$ time steps from now. The problem is to plan optimal trajectories for the N UAVs to cover the environment effectively. We define optimality by the notion of minimizing a cost function that represents the overall cost of the system at a particular time step.

Types of cells: A cell is 'good' if it the UAVs have not 'violated' its revisiting requirements, i.e., it has $r_{i,j} > 0$ at the current time step. A cell is 'bad' otherwise. Of the good cells, the lower its current revisit time, the more 'critical' it is. For visualization, we assign a particular color for each revisit time value. Bad cells are colored red and good cells are colored shades of blue, where a darker shade indicates a more critical cell.

Fig. 1. Environment representation example

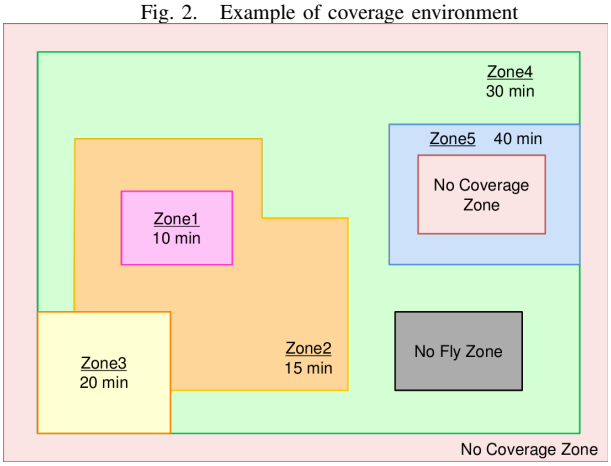


Since we need to efficiently cover this environment while satisfying these constraints, there are multiple objectives we need to satisfy:

1) Travel to a particular cell via an optimal path.
2) Have minimum number of 'bad' cells at any point of time in the environment.
3) Ensure that each UAV is assigned the best cell that will help satisfy the two aforementioned objectives.

In this way we model the state space as one with 5 degrees of freedom viz. (x, y, θ, v, t) with N UAVs. This could potentially lead to a joint state space of $5N$ degrees of freedom.

Fig. 2. Example of coverage environment



III. MOTION PRIMITIVES

In problems where analytic representations are not convenient, a useful step in problem formulation is to establish a sampling policy in order to avoid attempting to search the entire continuum. Beneficial sampling policies include those that cover a larger volume of the state space with fewer samples. It is natural to extend this concept from values of state space to paths to functions. Similar to the state space, the continuum of motions that are executable by the system can also be sampled to make computation tractable. In our case, we use a simple regular lattice aligning with the environment's (x, y) grid locations.

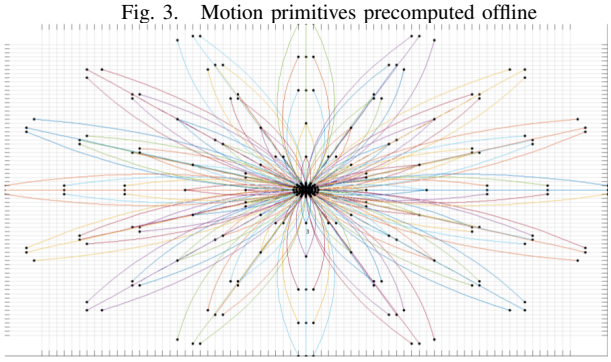
Assuming lattice state discretization outlined earlier, motion primitives are defined here to be the controls that connect roadmap vertices (states) and that are feasible motions. Motion primitives developed in such a way as to connect the vertices of the roadmap, sampled as a lattice in state space. By sampling regularly, the connectivity of this roadmap is completely specified with its control set. Any systematic graph search algorithm can be utilized to find the shortest path in the state lattice graph.

Given various restrictions on the UAV's linear acceleration, linear velocity, and angular velocity, we carefully devise kinodynamically feasible motion primitives assuming double integrator dynamics viz. $\dot{x}_{t+1} = \dot{x}_t + \ddot{x}_t \cdot \Delta t$. We assume discrete velocity values (2, 4, 6, 8 m/s), a trapezoidal angular velocity profile, a linear linear velocity profile, and a maximum angular velocity of 6 degrees per second. The advantage of assuming a trapezoidal velocity profile is to devise motion primitives that have a zero start and end angular velocity. In this way, we do not have to consider angular velocity in the state space.

IV. ALGORITHM

Our approach to this problem is two-fold. The first step is, given a particular scene in our environment (which consists of current revisit times for every cell and current UAV poses), to determine the best possible immediate goal locations on the frontier for each UAV. In this step, we account for three

Fig. 3. Motion primitives precomputed offline

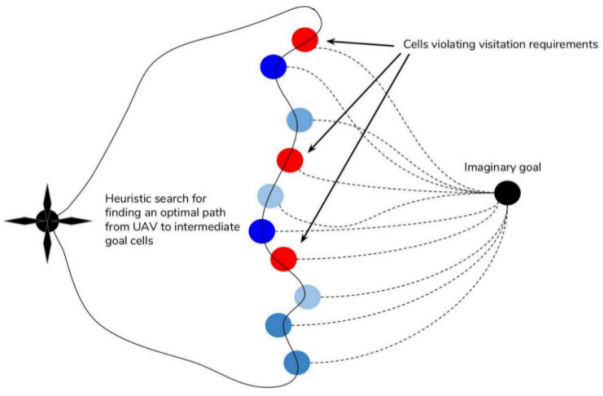


different restrictions: (1) The UAVs cannot fly into certain pre-designated no-fly zones, (2) We wish to select a location that ensures coverage of the most number of bad (and urgent) cells, and (3) We would like to minimize on travel time or travel distance. Considering travel distance for the last point, we perform this goal assignment via a multi-goal search for potential goals. For each UAV, we perform Algorithm 1.

Data: Discrete environment with M frontier cells marked, N UAV locations.
Result: Target goal (x_i, y_i) for each UAV i
for $i \leftarrow 1$ **to** M **do**
  for $j \leftarrow 2$ **to** w **do**
    Connect each cell j on the frontier to a single imaginary goal G;
    Assign a cost proportional to the cell's revisit time to the edge connecting to the imaginary goal;
  end
  Perform a 2D (x, y) search for each UAV i with its location as the start node and G as the goal node.
end

Algorithm 1: Multiple UAV Goal Assignment

Fig. 4. Goal assignment



In Algorithm 1, we perform a simple multiple-goal search to determine the next location each UAV should plan a path to. We first identify which cells are on the coverage frontier.

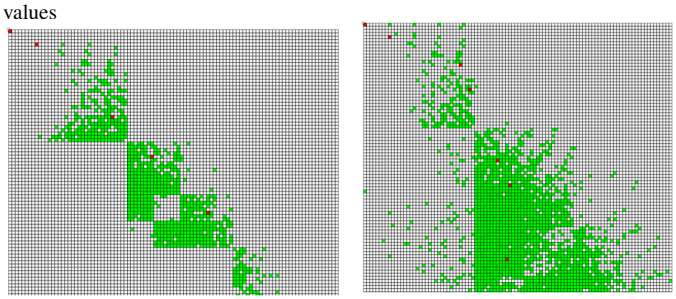
Data: Discrete environment with N UAV-goal pairs.
Result: Kinodynamically feasible persistent coverage plans.
Assign a random priority i to each UAV;
for $i \leftarrow 1$ **to** N **do**
  if $i == 1$ **then**
    continue;
  else
    Consider $(i-1)$-th UAV as a dynamic obstacle;
  end
  Plan a kinodynamically feasible path using precomputed motion primitives;
end

Algorithm 2: Coverage planner

Then we connect each frontier cell to an imaginary goal G via an imaginary edge, the cost of which is proportional to the current revisit time of that frontier cell. We then perform a 2D search from the UAV's current location to the imaginary goal. The effect of this is that the path chosen will determine which frontier cell the UAV should move to. The effect of adding the imaginary edges ensures that we optimize for the "best" frontier cell we would like the UAV to reach, and the search accounts for minimizing travel distance. Since the imaginary edge costs are proportional to the cell's revisit time, we can choose a proportionality constant depending on how much importance we want to give revisit time over the UAV's distance to the frontier cell.

In Algorithm 2, we implement *prioritized planning*. Decoupled approaches like this first design motions for the robots while ignoring any inter-robot interactions. Once these interactions are considered, the choices available to each robot are already constrained by the designed motions. If a problem arises, these approaches are typically unable to reverse their commitments. Therefore, completeness is lost. Nevertheless, decoupled approaches are quite practical, and in some cases completeness can be recovered. In our baseline implementation, we use a simple weighted-A* algorithm to plan feasible paths using the offline-precomputed motion primitives.

Fig. 5. Weighted-A* planner with expanded states for different epsilon values



V. CUSTOM VISUALIZATION

We also present the custom visualization tool written using the Qt SDK for graphics. Grid cell colors, UAV positions, headings, sensor footprints can be updated at each time step as "graphics items" in the Qt graphics framework.

Fig. 6. Sensor footprint approximated to a set of cells covered at any point of time in the environment.

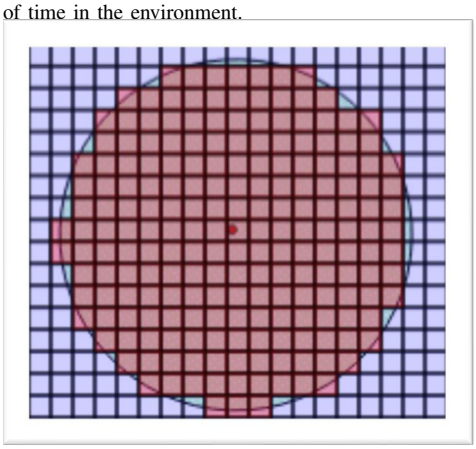
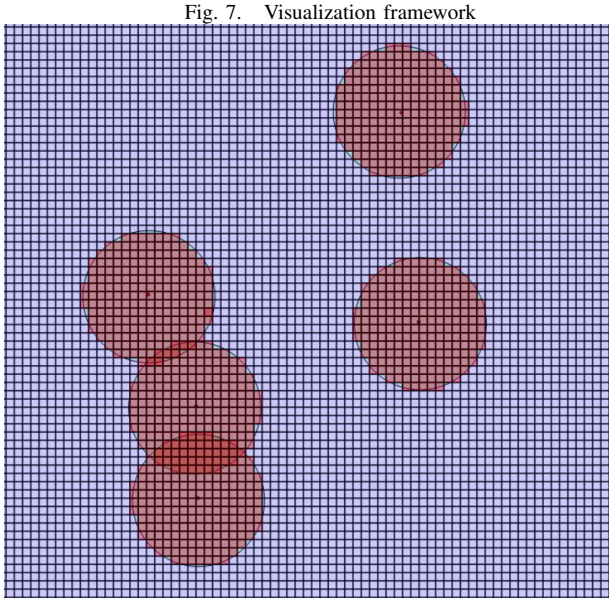


Fig. 7. Visualization framework



VI. SUMMARY AND FUTURE WORK

We present a baseline implementation of a multi-UAV coverage planner that produces kinodynamically feasible paths for multiple coverage zones. Future work involves learning from experience to produce better strategies for path planning.

ACKNOWLEDGMENT

The authors would like to thank the Federation of Indian Chambers of Commerce and Industry (FICCI) for supporting this work.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[2] F. Yasutomi, M. Yamada, and K. Tsukamoto, "Cleaning robot control," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pp. 1839–1841, IEEE, 1988.

[3] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Uniform coverage of automotive surface patches," *The International Journal of Robotics Research*, vol. 24, no. 11, pp. 883–898, 2005.

[4] Z. L. Cao, Y. Huang, and E. L. Hall, "Region filling operations with random obstacle avoidance for mobile robots," *Journal of Robotic systems*, vol. 5, no. 2, pp. 87–102, 1988.

[5] B. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures.," in *Icaps*, 2012.

[6] L. Xu, "Graph planning for environmental coverage," 2011.

[7] J. Butzke and M. Likhachev, "Planning for multi-robot exploration with multiple objective utility functions," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3254–3259, IEEE, 2011.

[8] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1093–1099, IEEE, 2015.

[9] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot persistent coverage with stochastic task costs," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 3401–3406, IEEE, 2015.

Segmentation of Humans from LiDAR Point Clouds Using Visual Pose Estimation

Gaini Kussainova[1], Luis E. Navarro-Serment[2], Martial Hebert[2]

***Abstract*— Human segmentation from point clouds usually requires high computational budgets that result in slow processing speeds. This paper proposes an algorithm for fast, reliable and computationally inexpensive segmentation of humans from point clouds that is suitable for common robot configurations comprising a camera and a 3D LiDAR. The algorithm takes advantage of the detections from a visual pose estimation process that detects key points from the human body–like limbs and facial key points–from images and uses them to identify and cluster the points in the cloud that represent humans. The algorithm consists of two basic steps. First, the 3D points from the cloud are projected onto the camera's image plane and then filtered according to their proximity to the regions in the image labeled as humans by the pose estimation process. Then, the filtered 2D body coordinates are projected back to the 3D space and further processed, adding or discarding other points nearby depending on their geometrical relationship with the original set.**

The proposed approach relies heavily on state-of-the-art algorithms for visual detection of key points from the human body, and does not require significant additional computational budget, making it suitable for real-time operation and for operation with sparse point cloud data.

I. Introduction

Accurate automatic human segmentation is an important problem in computer vision due to its applicability to different problems, such as video surveillance, understanding of crowd behavior, motion tracking, and activity recognition. It is particularly relevant for socially-aware robots because they must interact, navigate, and act in spaces occupied by people.

Achieving accurate human segmentation for robotic applications is not a trivial task due to occlusions, changes in the point of view, moving objects, changes in illumination, and others [1].

There is a significant number of human segmentation algorithms for both images and point clouds. Image segmentation algorithms have been used extensively in all of their forms: clustering, region growing, and image-domain techniques. Although they provide good segmentation results, they typically do not provide a complete description of the spatial layout of objects [2]. Conversely, LiDAR sensors provide the ability to directly analyze shapes, sizes, and distances in Euclidean space. For this reason, it is desirable to implement perception systems based on 3D point clouds. However, the robust and accurate detection of humans from point clouds is difficult, mainly because the number of points representing a person decreases with the distance from the sensor, and consequently reduces the information available to establish an accurate identification. This problem is compounded by the use of low-resolution scanners, which have become more prevalent because of their lower cost and smaller size.

In this work, we propose a fast, reliable, and computationally inexpensive algorithm for human segmentation from point clouds, which was motivated by our application on a mobile ground robot that operates among people. The algorithm is able to accurately segment humans without the use of appearance-based descriptors. To minimize the computing budget required, the algorithm takes advantage of the detections generated by a visual pose estimation module, which uses a monocular camera. These detections are sets of key points that indicate the locations in the image of human body joints. This module consumes the largest part of the computing budget; our algorithm benefits from this effort and performs local searches on very specific areas in the point cloud guided by the pose detections. These searches are computationally inexpensive and allows the segmentation algorithm to piggyback on the pose estimation system with little additional processing time.

The paper is organized as follows: the next section presents the related work describing segmentation techniques overall and human segmentation methods. Then, section III contains the detailed explanation of our approach. Section IV presents the evaluation of our algorithm and section V concludes the paper and discusses the future modifications to the algorithm.

II. Related Work

Although image segmentation is an important task in computer vision, and it has been implemented in multiple applications [4], segmentation techniques from monocular images need to perform elaborated processing to generate geometric descriptions in 3D that can improve their performance significantly. Conversely, 3D point clouds directly provide valuable geometric information, such as location and 3D distances, that simplify the task of differentiating individual objects. However, point cloud segmentation is not a trivial task due to measurement noise, sparseness and uneven density in the data. Point cloud segmentation can be edge-based, region-based and model-based algorithms. Edge-based algorithms analyze the shape to find the boundary. They are fast, but not very accurate in segmenting noisy, and unevenly distributed

[1]Gaini Kussainova is a research scholar of Robotics Institute, Carnegie-Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA gaini.kussainova at nu.edu.kz

[2]Luis E. Navarro-Serment and Martial Hebert are a project scientist and Faculty (respectively) at the Robotics Institute, Carnegie-Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA lenscmu@ri.cmu.edu, hebert@ri.cmu.edu

points. Region-based methods cluster neighboring points if they satisfy some criteria. Although region-based methods segment the noisy data more accurately, they are more susceptible to over- or under-segment the data. The approach described in [5] made a region-based point cloud segmentation, where user sets the desired level of abstraction, local threshold. Such algorithms are very sensitive to the user input. Model-based algorithms use geometric shapes, mathematical representation, to group points in one region [6]. The algorithm used in [7] made the segmentation of sparse point clouds by implementing the region-growing segmentation with the use of geometric relation between neighboring points and point clouds edges distribution analysis, and then used K-means clustering method to refine the segmentation. 3D semantic labeling in [8] was implemented with the use of deep neural networks; they created 2D views from a 3D scene, made segmentation on 2D plane with the use of deep networks, and projected the results of 2D segmentation back to the original 3D space.

Approaches that segment particularly people can be implemented effectively with the use of human body knowledge. The approach in [13] made the human parts-based detection and segmentation with the use of human instance structure. The output of face detectors as input was used in [9] for human segmentation. Their algorithm iteratively apply grabcut by analyzing the color model and increase the 'human' labeled region. However, the use of only color model is not enough to make the accurate segmentation. In [10] the human segmentation from multi-modal visual cues was implemented using RGB-depth-thermal dataset. Although a lot of segmentation and tracking of human methods use full and complex 3D modeling of human being, they are too slow and cannot be used in real time or require additional camera/scene setup [11].

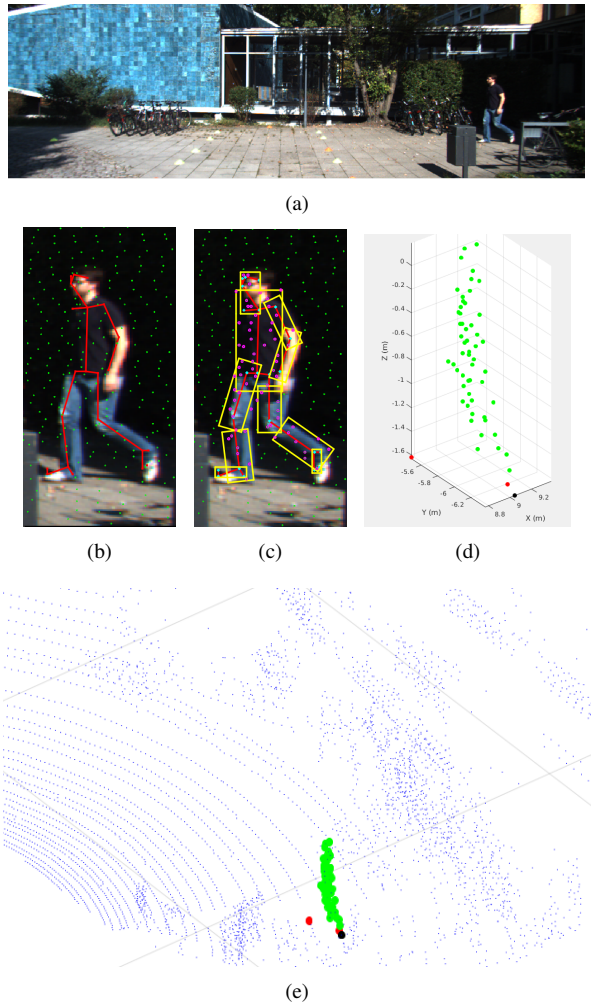


Fig. 1. The intermediate steps of the algorithm: (a) is the original image, (b) the cropped image with skeleton points (red) and projected 3D point cloud data (green) on image plane, (c) bounding boxes (yellow) around every body region and projected 3D point cloud data (magenta) labeled as human after the 1 stage of the algorithm, (d) the zoomed output of the algorithm: black points - false negatives, green points - true positives, red points - false positives, (e) the output of the algorithm: blue points - true negatives.

III. The Proposed Approach

In this section we describe the implementation details of the proposed approach. The algorithm takes 3D point cloud data and 2D pose estimation key points as input, and returns a 3D point cloud whose points are labeled as human or not-human as an output. Fig. 1 shows the intermediate and final results of the proposed algorithm. The pseudo code description of this approach is shown in Algorithm 1.

To perform visual pose estimation we use OpenPose [12], a state-of-the-art system that detects human key points in real time. It detects the location in the image of 25 main body key points for all the humans present in the scene (within a certain distance). Our approach uses human pose estimations because they provide a good indication of the location of humans on the image, and also of the regions where LiDAR measurements are most likely to be found. Moreover, pose estimations generate image coordinates for most body joints, which allows obtaining seed points covering the most salient body parts. With the use of pose estimations as input to our algorithm, we can make a reliable human segmentation by the analysis of only geometrical relationships between points. As a result, the algorithm is fast and requires minimal additional computational power.

The algorithm starts by projecting the point cloud onto the image plane of the camera (we assume that the camera and the LiDAR have been extrinsically calibrated). This is done to examine their location with respect to the detections from visual pose estimation in the same plane. The algorithm then inspects the estimates of pose of the human skeleton and includes only the points that either belong to a limb or lie in a close proximity to it; this prevents the algorithm from analyzing irrelevant points. This inspection is done by creating a bounding box around the human skeleton, and expanding it from all four sides, as shown in Fig. 1-(c). The width and height of the human bounding box was increased

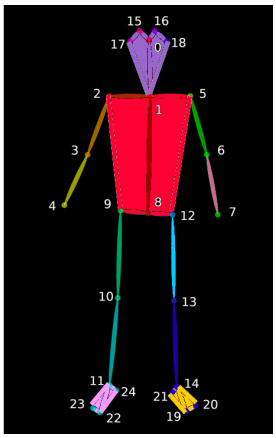


Fig. 2. The 25 human body keypoints detection and 12 regions used in the algorithm illustrated by different colors.

by quarter of a body width and quarter of a body height respectively from both sides, resulting in expansion of a bounding box's height and width by half the human height and width. Such expansion assures that the algorithm does not discard human points before conducting further analysis. The removal of irrelevant points decreases the number of points to process, which in turn reduces the processing time.

The next step is the initialization of regions of interest. If a point is inside one of the region of interest, it is labeled as a 'human' for further processing and analysis. We used 12 regions of interest: head + neck, torso, right and left arms before elbow, right and left arms after elbow, right and left legs before knee, right and left legs after the knee, right and left foots. The regions are illustrated in Fig. 2. We decided to make arms (legs) before and after the elbow (knee) two different regions, because we use the bounding boxes around every region, and bend of arm or leg leads to a bigger bounding box and more false positive human points. When OpenPose fails to detect a key point, the algorithm ignores it and creates the region from the only the existing detected points. However, if a region consists of only two points, such as arms and legs, and one of the body key point was not detected, the algorithm does not consider that region. The algorithm expands the bounding box around every joint horizontally and vertically by the half of a maximum horizontal and vertical distances between points in the region. The expansion is carried out to ensure that all potential 'human' points will be analyzed.

After all the bounding boxes have been created, the points projected onto the image plane are examined to determine whether they are located in one of these regions and marked as 'humans' for further analysis (line 7 - line 10).

At this point, the algorithm has generated an initial set of points that are candidate elements of the 'human' set using only geometrical relationships between points in the image plane. In the next step, the algorithm examines the set of points from the previous step, but now back in the original 3D space. To compute the inverse mapping from image plane to Euclidean space, we need to observe constraints to resolve the uncertainty in distance from the camera. To this end, the ground plane is found using the orientation specified by a normal reference vector obtained from extrinsic calibration. To find the coordinates of the human skeleton in 3D space, we use the projection of the image coordinates of the key points corresponding to the skeleton's feet to 3D point and calculate the location of the intersection with the ground plane. The geometric constraint is that the z coordinate of the feet which is closest to the ground plane is used to determine the location on the ground plane. Then, the points that are located outside of a 1.5 m radius from the human skeleton in 3D are discarded. To decrease the number of false positives, points belonging to the ground are discarded if they are located 0.75 meter away from feet.

In the final step the goal is to remove all points that were mistakenly labeled as 'human' in the previous steps by inspecting the distances between points in every region. The main idea was that points in the same region should be close to each other. Therefore, the algorithm discards points that are too distant from the other points in the same region. This is accomplished by calculating the mean distance of point to all the other points in the same region (line 28 - line 33) and the mean distance between every pair of points in the region (line 34 - line 38). These values are calculated for every point and every region, and compared for the final labeling.

IV. Results

We have performed experiments using the KITTI dataset[1], which contains synchronized, calibrated, and timestamped data from multiple sensors collected from a moving platform in urban environments [3]. There is a calibration file with all the intrinsic and extrinsic calibration parameters for every sequence. The dataset includes images and point clouds captured by two high-resolution color and grayscale video cameras, and a Velodyne 64 laser scanner with a 360° field of view. It also contains manually annotated ground truth bounding boxes for five classes: cars, trams, trucks, pedestrians and cyclists.

The algorithm was implemented using MATLAB. To test our approach we annotated by hand 65 examples of 3D point clouds, which were selected to represent various combinations of human pose, distance and orientation to sensors, while retaining the ability of OpenPose to generate accurate detections. The mean, maximum, and minimum distance from the sensor to the human from our test set were 8.99 m, 11.85 m, and 6.1 m respectively. The standard deviation of distances for 65 selected examples is 1.85 m. Fig. 3 illustrates several samples and algorithm outputs.

We evaluated the performance of the algorithm using precision and recall as performance metrics. The precision measures the percentage of correctly predicted points

[1]http://www.cvlibs.net/datasets/kitti/index.php

Algorithm 1: Human Segmentation Algorithm

Input : $P_k = \{p_1, p_2, .., p_k\}$ - point cloud, where $p_i =< x_i, y_i, z_i >$
$S_m = \{s_1, s_2, .., s_m\}$ - pose estimations where $s_j =< u_j, v_j >$

Output: $HS_n = \{hs_1, hs_2, .., hs_n\}$ segmented point cloud, where $hs_l =< x_l, y_l, z_l >$

Parameters: $PR_k = \{pr_1, pr_2, .., pr_k\}$ - point cloud projected on image plane, where $pr_j =< u_j, v_j >$
boundBox - $[x_{min}, x_{max}, y_{min}, y_{max}]$ - skeleton bounding box
$R = \{R_1, R_2, ., R_12\}$ - bounding boxes around every body part
factor - factor of region expansion
$H = \{H_1, H_2, ., H_n\}$ - human labeled points in image plane; $H_i =< u_i, v_i >$

Functions : PROJECT(P_k, calibrationMatrices) - makes homogeneous transformation of point clouds onto image plane

Begin:
READ skeleton file
removeUndetectedSkeletonPoints(S_m)
PR_m = PROJECT(P_k, calibrationMatrices)
crop(PR_m, skeleton bounding box)
expandBoundingBoxes(R, factor)
for p_i ***in*** PR_m **do**
  for r_j ***in*** R **do**
    if p_i *belongs to* r_j **then**
      $H_p[r_j].append(p_i)$
    end
  end
end
fitGroundPlane($referenceVector$)
x_{feet_3D}, y_{feet_3D} = PROJECT(calibrationMatrices, x_{feet_2D}, y_{feet_2D}, $mean(z_{ground})$)
meanPoint(H_p)
meanRegion(R)
for p_i ***in*** PR_m **do**
  for r_j ***in*** R **do**
    if $meanPoint(p_i) < meanRegion(r_j)$ *AND* $p_i inr_j$ *AND* $dist(p_i, < x_{feet\_3D}, y_{feet\_3D}, z_{feet\_3D} >) < 1.5$ **then**
      $HS_n\{p_i\} = human$
    end
  end
end
return HS_n
End

meanPoint(p)**:**
for p_i ***in*** R **do**
  mean = mean(dist(p_i, p))
end
return mean

meanRegion(R)**:**
for *all combinations*(p_i, p_j) ***in*** R **do**
  mean = mean(dist(p_i, p_j))
end
return mean

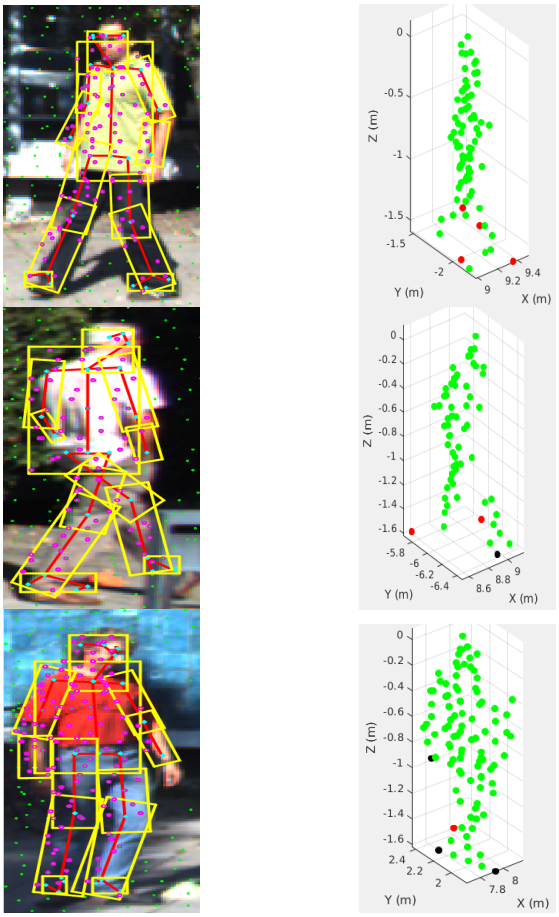


Fig. 3. The output of the algorithm for different samples: the left column shows the cropped image with skeleton points (red), projected 3D point cloud data (green) on image plane, bounding boxes (yellow) around every body region and projected 3D point cloud data (magenta) labeled as human after the 1 stage of the algorithm; the right column shows the output of the algorithm: green points - true positives, black points - false negatives, red points - false positives.

from all the points labeled as 'human'. The recall metric represents the percentage of all the 'human' points that were correctly detected as 'human'.

The mean of the precision and recall values for our algorithm is 94.45% and 97.55% respectively in our test dataset. These and other statistics are presented in Table 1.

The hardware requirements for the proposed approach are mainly imposed by OpenPose: an NVIDIA graphics card with available 1.6 GB and 2.5 GB of free RAM memory[12].

	MEAN (%)	MAX (%)	MIN(%)	STDEV (%)
Precision	94.45	100	85.25	2.84
Recall	97.55	100	89.09	2.65

TABLE I
TABLE 1. SEGMENTATION PERFORMANCE.

V. Conclusion and Future Work

This paper described a fast and computationally inexpensive algorithm for human segmentation from LIDAR point clouds. The algorithm relies on OpenPose's capabilities for human pose estimation and uses the 3D point cloud to consider geometric relationships between the body regions, resulting in accurate and robust segmentation.

However, this reliance results in ungraceful degradation of segmentation performance if the pose estimation system performs poorly. Therefore, it is recommended to investigate mechanisms to mitigate the negative impact in such situations.

References

[1] A. Hernndez, M. Reyes, S. Escalera and P. Radeva. "Spatio-temporal grabcut human segmentation for face and pose recovery," in *Proc. of 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 33-40, 2010.

[2] A. Anand, H.S. Koppula, T. Joachims and A. Saxena. "Contextually guided semantic labeling and search for three-dimensional point clouds," *The International Journal of Robotics Research*, pp. 19-34, 2013.

[3] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, pp. 1231-1237, 2013.

[4] P. Arbelaez, M, Maire, C. Fowlkes and J. Malik. "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 898-916, 2011.

[5] A. Dimitrov and M. Golparvar-Fard "Segmentation of building point cloud models including detailed architectural/structural features and MEP systems," in *Automation in Construction.* pp. 32-45, 2015.

[6] A. Nguyen and B. Le. "3D point cloud segmentation: A survey," in *Proc. of the 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 225-230, 2013.

[7] M. Li and D. Yin. "A fast segmentation method of sparse point clouds," in *Proc. of the IEEE Control And Decision Conference (CCDC)*, 2017.

[8] A. Boulch, B. Le Saux and N. Audebert "Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks," in *Proc. of the Eurographics Workshop on 3D Object Retrieval 3DOR*, 2017.

[9] A. H. Vela. "Pose and face recovery via spatio-temporal grabcut human segmentation," M.Sc. Thesis, Universidad Autonoma de Barcelona, 2010.

[10] C. Palmero, A. Clapés, C. Bahnsen, A. Møgelmose, T. Moeslund and S. Escalera. "Multi-modal RGB-depth-thermal human body segmentation," *International Journal of Computer Vision* 118.2 (2016): 217-239.

[11] N. T. Siebel and S. Maybank. "Fusion of multiple tracking algorithms for robust people tracking," *European Conference on Computer Vision*, pp. 373-387, April, 2002.

[12] Z. Cao, T. Simon, S.-E. Wei and Y. Sheikh. "Realtime multi-person 2d pose estimation using part affinity fields," *CVPR,* 2017.

[13] V, Vineet, J. Warrell, L. Ladicky and P. Torr. "Human Instance Segmentation from Video using Detector-based Conditional Random Fields," in *Proc. of the British Machine Vision Conference*, pages 80.1-80.11. BMVA Press, September, 2011.

Evaluating Accuracy of DSRC GPS for Pedestrian Localization in Urban Environments

*Aidan Lakshman[1] lakshman@knights.ucf.edu
*Viraj Parimi[2] parimi15068@iiitd.ac.in
Stephen F. Smith[3] sfs@cs.cmu.edu
Isaac Isukapati[3] isaack@cs.cmu.edu

***Abstract*— Dedicated Short-Range Communications (DSRC) are a standard of communications designed for vehicle-to-vehicle and vehicle-to-infrastructure communication. This technology offers significant promise for improving pedestrian safety, as it allows pedestrians to communicate directly with nearby infrastructure elements and vehicles, and offers more precise GPS localization. Our work shows the current standard is too imprecise at low speeds, and as such cannot be used for pedestrian localization. We propose a method to increase the accuracy of DSRC GPS readings at low velocities and to increase precision by utilizing both iPhone and DSRC GPS readings.**

I. Introduction

DSRC was developed in 1999 by the United States Federal Communications Commission as a short range, low-latency, line of-sight wireless data transmission standard designed for interactions between vehicles and infrastructure in urban environments. DSRC messages use the Basic Safety Message (BSM) for safety-related applications, which is described below. Non-safety related information is communicated with the other message formats, which can include toll collection data, digital maps, intersection signal status, and other information. Some messages can also provide more detailed information on intersection geometries and positioning by using known coordinates of DSRC intersection nodes in conjunction with differential GPS algorithms [2].

A. BSM Message

The main message standard used in positioning without the use of DSRC-enabled intersections as reference points is the Basic Safety Message (BSM). The BSM is comprised of a set of data frames, each containing data elements or data frames. The elements included in the BSM are defined in the ASN.1 coding standard, and can include information on vehicles, weather conditions, road conditions, positioning, and other information. All BSMs include a set of basic elements, denoted as Part 1. This includes basic positional information, motion information (such as speed, heading, etc.), brake system status, and vehicle size. All remaining information is optional, and can be included in Part 2 of the BSM. The BSM is encoded as a binary file, with each data element allocated a specific number of bytes (as described in the ASN.1 coding standard[3][4]). Messages are sent to DSRC-enabled devices, which decode the message into its components. As each data element has a maximum number of bytes allocated, the granularity of the data is determined by the number of bytes it has access to in the message.

B. Goals

Mobility-impaired pedestrian often have greater needs for intersection navigation than average pedestrians. The long-term goal of our work is to improve the safety of mobility-impaired pedestrians when crossing busy intersections. In order to implement a more robust system for this community, we need to be able to accurately communicate their needs to surrounding infrastructure and vehicles. Currently, pedestrians have no way to communicate with cars or intersection controllers other than Accessible Pedestrian Signals, which have been previously proven to be ineffective for increasing safety[1]. In addition, cell phone GPS has a reported accuracy of 10 meters, which is not precise enough to measure the position of a pedestrian within a typical intersection. The primary intention of this paper is to experiment with implementing a DSRC-enabled device with which a pedestrian can communicate with nearby DSRC-enabled nodes to improve their safety when navigating an intersection. Current DSRC enabled devices report a GPS accuracy of less than one meter, which would theoretically provide enough precision to ensure safer pedestrian mobility in urban settings.

II. Experiment Details

The BSM message allocates four bytes for each coordinate value (latitude and longitude), expressed in 1/10th microdegrees. Our experiment seeks to improve the ability of pedestrian-infrastructure communication by measuring the accuracy of localization of DSRC devices and normal GPS devices found in cell phones. iPhones Location Manager allocates 8 bytes for each coordinate value, allowing for precision to the femtodegree (10^{-15}).

We used the Arada Systems LocoMate ME Mobile V2X Sleeve to allow our iPhone to receive BSM messages for positioning and recorded these alongside GPS readings from the iPhone's onboard GPS. GPS signals are received by an Arada Systems DSRC-enabled device (denoted as the "transmitter"), and are then sent to the Mobile Sleeve (denoted as the "receiver") as a BSM message, which is passed to the iPhone. The two Arada devices were kept at a distance of 6-18 inches throughout all tests.

*A. Lakshman and V. Parimi are co-authors

[1]A. Lakshman is with Department of Mathematics, University of Central Florida, and is funded through the NSF.
[2]V. Parimi is with the Department of Computer Science, Indraprastha Institute of Information Technology, and is funded through FICCI.
[3]S. Smith and I. Isukapati are with the Robotics Institute, Carnegie Mellon University

We used an iPhone 7 for testing and performed trials at several urban environments in Pittsburgh to test performance under different settings.

We use the track test to test the accuracy of the GPS devices in an environment with no nearby buildings or obstructions. This test should indicate the best possible accuracy for each device. Intersection tests involved walking in a circle around all crosswalks in an intersection. Every path was taken in the direct center of each crosswalk around the intersection. All attempts were made to walk at a constant rate of approximately 1 m/s, but due to traffic and signal timings our actual walking speed fluctuates between 0.75 and 1.25 m/s. Different intersections were tested to experiment with different building heights and urban canyons.

The roadside test involved walking alongside a road, making a 90 degree turn, and walking down a second road. This was to test the performance of the GPS units when the pedestrian took a path similar to that of a vehicle.

Finally, the driving test tested the performance of the GPS units in high speed setting. This was designed to measure the normal performance of the DSRC GPS, as DSRC is specifically designed for vehicle-to-vehicle communication at high speeds. This test involved taking measurements while driving at speeds of 40-60 miles per hour. The transmitter and receiver were both placed on the front dashboard to allow clear communication with as little obstruction as possible. The two devices were kept approximately 12 inches apart for the duration of the test, and maintained a constant relative distance.

In all tests, both the iPhone GPS and the DSRC messages were recorded for the same length of time.

The iPhone GPS reliably updates once every second. The Arada DSRC Sleeve attempts to update as often as ten times per second, but in our experiments the update rate ranges from 8.82 readings per second to 0.16 readings per second. This large range in update rates is attributable to messages being lost while transmitted between the two Arada devices.

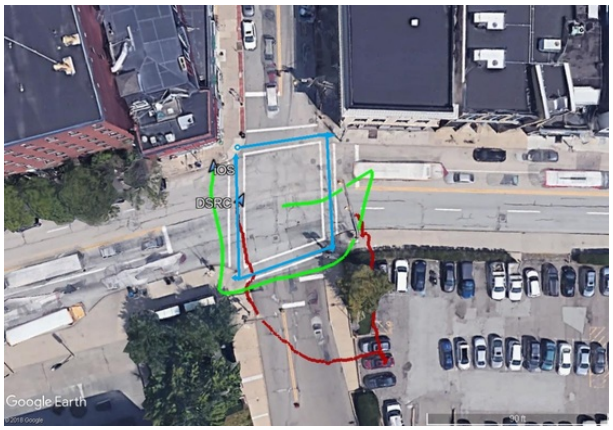


Fig. 1: Craig St. and Forbes Ave.

III. Results and Analysis

Diagrams of the intersections tests are found in Figure 1 and in the Appendix. Red lines represent the coordinate readings from the DSRC information, and Green lines represent the coordinate readings from the iPhone 7. The actual path taken is denoted in blue (black for the track test).

In the track test as well as Avenue and Regular Canyons, the iPhone GPS performs significantly better than the reported 10 meter accuracy. Actual accuracy of the iPhone in urban environments is around 1-2 meters.

In contrast, the DSRC device reported accuracy significantly worse than its claimed <1 meter accuracy. In the best tests, the DSRC GPS accuracy was 5 meters, and in the worst case around 200 meters. As most intersections are less than 15 meters wide, these readings show the DSRC granularity is too large for localization in urban intersection environments.

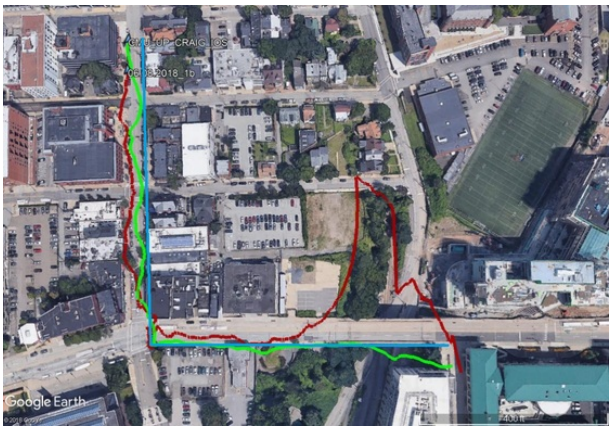


Fig. 2: Roadside Test

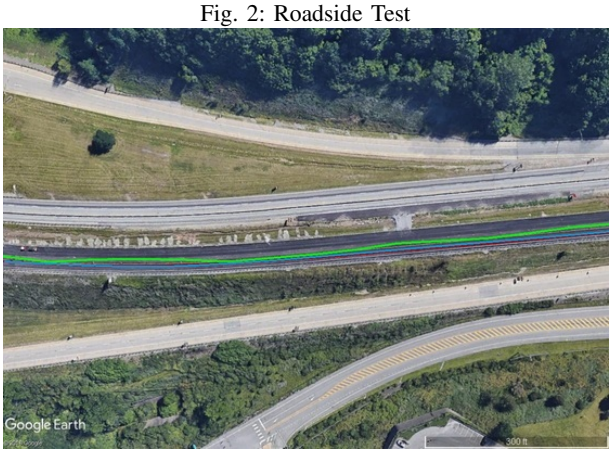


Fig. 3: Driving Test

Figure 3 show the results of the driving test. At higher speeds, DSRC GPS reports accuracy of under 0.5 meters, compared to 2 meter accuracy with iPhone. This either suggests that DSRC GPS is utilizing speed as a correction measure, or that DSRC is attempting to ”push” measurement readings onto roads with the assumption that vehicles are utilizing the communication protocol.

Figure 2 shows the roadside test. DSRC performed no better in this trial than in the original pedestrian intersection tests, meaning DSRC is likely not correcting measurements by assuming all readings are taken from roads. This leads to the conclusion that speed is a major factor in DSRC GPS readings. However, this reliance on speeds leads to overfitting in pedestrian environments, as pedestrian

navigation is much slower than vehicle navigation.

IV. Conclusions and Future Work

Our findings show the DSRC communication is currently too inaccurate for use in localization of pedestrians. The two main problems in DSRC communication is the highly variable refresh rate, and the inaccuracy of measurement at low speeds.

DSRC packet loss has been previously been shown to be volatile in research studies involving truck platoons. Gao et. al. showed that numerous factors can contribute to packet loss with broadcast messages (such as BSM). Line of sight obstruction as well as latency caused by lower level components of the DSRC transmitter can have a detrimental effect on the data delivery ratio[5]. In future tests, we will position the transmitter and receiver to minimize any line of sight obstructions, and will look into the hardware of the transmitter/receiver to maximize the number of GPS readings per second.

In addition, we plan to isolate the internal corrections using speed to help predict for position. If we can access GPS data before they are adjusted and cast into a BSM message, we can achieve greater accuracy, as the low differences in speed are causing inaccuracies in position measurements.

We plan to implement these solutions and retest the DSRC sleeve to attempt to obtain accuracy of at least 5 meters. Many past research projects have detailed algorithms to combine inaccurate GPS readings to obtain higher precision than a single GPS receiver. Schrader showed that in the best case, multiple GPS devices used in conjunction can yield up to a 27% improvement in precision compared to a single GPS unit[6]. Trinklein performed an experiment using two clusters of GPS receivers each with 3 meter accuracy. The clusters were kept at a constant distance of 4.5 meters, but the system as a whole was mobile. Trinklein's algorithm was able to exploit the constant relative distance between the two clusters to achieve approximately 1 meter accuracy[7]. Hedgecock et. al. developed an algorithm to obtain position with accuracy of 15 centimeters given two or more GPS receivers with 2.5 meter accuracy. This algorithm utilized raw GPS data and satellite positioning information to correct errors due to satellite velocities and temporal differences in readings[8].

We plan to implement a variation of these algorithms to use both the iPhone GPS and DSRC GPS signals together to obtain precision of under 1 meter, allowing for precise pedestrian localization in small-scale urban environments, such as intersections.

V. Appendix

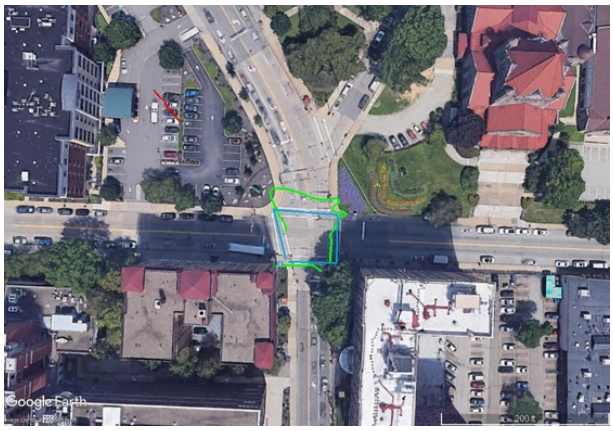


Fig. 4: Centre Ave. and Aiken Ave.

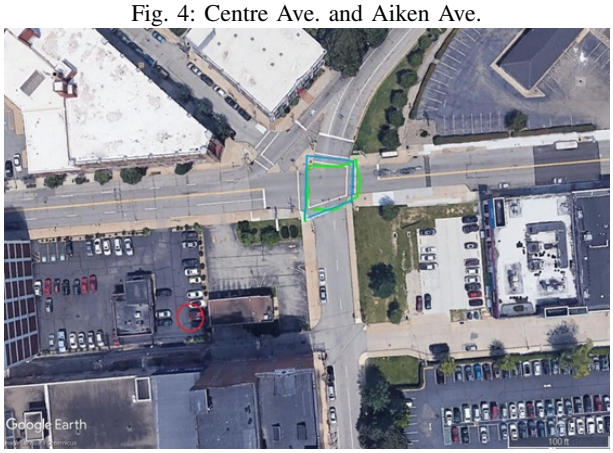


Fig. 5: Baum Blvd. and Euclid St.

Fig. 6: Baum Blvd. and Liberty Ave.

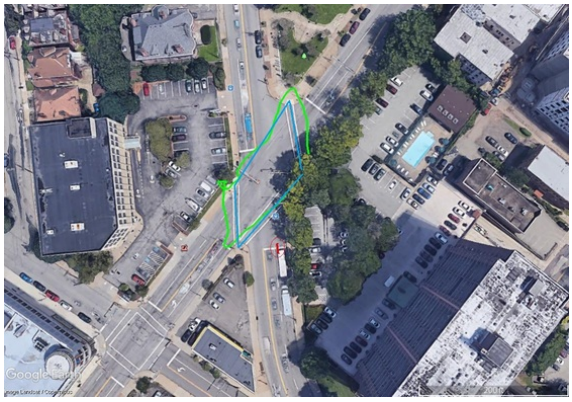


Fig. 7: Baum Blvd. and S. Negley St.

Fig. 8: Centre Ave. and Cypress St.

Fig. 9: Centre Ave. and Graham St.

Fig. 10: Centre Ave. and S. Highland Ave.

ACKNOWLEDGEMENTS

Thank you to Dr. Isaac Isukapati, Dr. Stephen Smith, Dr. Zack Rubinstein, and Joseph Zhou for their mentorship and support throughout this project. Also, thank you to Rachel Burcin and John Dolan for organizing the CMU RISS program. Their constant enthusiasm and encouragement have been an inspiration and a motivation for our efforts this summer. Finally, thank you to NSF and FICCI for their financial support throughout this project.

REFERENCES

[1] Chen-Fu Liao, Using a Smartphone Application to Support Visually Impaired Pedestrians at Signalized Intersection Crossings, in Journal of the Transportation Research Board, Washington, D.C., 2013. doi: 10.3141/2393-02

[2] G. Naik, J. Liu and J. J. Park, "Coexistence of Dedicated Short Range Communications (DSRC) and Wi-Fi: Implications to Wi-Fi performance," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9. doi: 10.1109/INFOCOM.2017.8057214

[3] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States," in Proceedings of the IEEE, vol. 99, no. 7, pp. 1162-1182, July 2011. doi: 10.1109/JPROC.2011.2132790

[4] International Transportation Union, "OSI networking and system aspects Abstract Syntax Notation One (ASN.1)," in Series of ITU-T Recommendations, Series X.690, July 2002.

[5] Song Gao, Alvin Lim, David Bevly, "An empirical study of DSRC V2V performance in truck platooning scenarios," Digital Communications and Networks, Volume 2, Issue 4, 2016, Pages 233-244. doi: https://doi.org/10.1016/j.dcan.2016.10.003.

[6] Schrader, Daniel Kenneth, "Combining Multiple, Inexpensive GPS Receivers to Increase Accuracy and Reliability" (2013). Open Access Theses. Paper 76.

[7] Trinklein, Eddy H., "Post processing of multiple GPS receivers to enhance baseline accuracy", Master's Thesis, Michigan Technological University, 2011. http://digitalcommons.mtu.edu/etds/414

[8] Will Hedgecock, Miklos Maroti, Janos Sallai, Peter Volgyesi, and Akos Ledeczi. 2013. High-accuracy differential tracking of low-cost GPS receivers. In Proceeding of the 11th annual international conference on Mobile systems, applications, and services (MobiSys '13). ACM, New York, NY, USA, 221-234. DOI: https://doi.org/10.1145/2462456.2464456

FPGA Acceleration for High Dimensional Inverse Kinematics

Yike Li[1], Lu Li[2], Howie Choset [3]

***Abstract*— A robot has to solve inverse kinematics problem in order to reach the desired target. The numeric solution for solving inverse kinematics of a high degree of freedom robotic arm is computationally expensive and resource demanding, especially in a real-time closed-loop control scenario. We propose an acceleration method that uses FPGA to parallel compute the Jacobian method algorithm. Also, we develop the system with high-level synthesis tool, so we can use programming language C++ instead of low-level hardware description language (HDL), which makes it easier to reorganize the design for the further application. The proposed method is implemented and verified on a low-cost heterogeneous FPGA SoC. The hardware implementation of the accelerator is described and method performances are analyzed.**

I. INTRODUCTION

To complete certain tasks, like feeding disabled people or grasping a package on the goods shelf, industrial robotics or other robot manipulators need to follow an optimized trajectory, so that they can reach the target fast while avoiding collision with obstacles. Inverse Kinematics problem is the transformation of position and orientation of a manipulator end-effector from Cartesian coordinates to joint coordinates. [1] Robot manipulator uses inverse kinematics to reach the desired end-effector position.

Many of these robots have multiple joints, the redundant DOF give multi-solution in robotics joint space, which is helpful for robot arms to avoid both internal problems and external obstacles, such as the ability to perform the same task when one of the joint angles is broken or using different trajectory planning to avoid the obstacles that occurred in the robot working place.

The one to many relationships between task coordinates and joint coordinates endow the robot with greater dexterity and flexibility, however, the solution for redundant inverse kinematics problem is infinite. [2] Therefore, solving high dimensional inverse kinematic problems in an optimized manner has been an increased research interest in the robotic field.

To solve the inverse kinematics, there are two types of methods, analytically and numerically. [3] Analytical solutions exist only for some special robots that have geometry symmetric property or other inner structural connections like paralleling with each other. Within these constraints, the DOF can be reduced and the solution can be analytically expressed. However, in some cases, the explicit analytical solution does not exist, in the absence of the analytical solution, the numerical techniques are alternative options for solving inverse kinematics.

There are plenty of methods to solve IK numerically, such as quasi-Newton and conjugate gradient methods [1] [4], neural net and artificial intelligence methods [5] [6], cyclic coordinate descent methods [1], pseudoinverse methods [7], Jacobian transpose methods [8] and the Levenberg-Marquardt damped least squares methods [9]. Each of these methods needs a large amount of computation to get the final results, and the procedures usually involve large amounts of matrix operations such as matrix multiplication or matrix inversion, results in that the process of calculating IK of the end-effector spend much CPU time, which will affect the motion response in robot operation. Therefore, solving this problem in a fast and efficient way becomes an important issue.

A field-programmable gate array (FPGA) is an integrated circuit of gate level design that can be programmed according to the designer's special needs. The FPGA is similar to an application-specific integrated circuit (ASIC) in functional aspect, they both use to specify the configuration to perform a certain task, however, ASIC is more costly in terms of the manufacturing since a single ASIC product has unchangeable configuration and can only be used in one scenario. FPGA, on the other hand, is convenient to be reprogrammed according to different demands.

FPGAs are used to create custom hardware circuits, such as repeatable data processing functions that would be resource intensive when performed by micro-controllers or local central processing units (CPUs). Also, FPGA can be used to perform hardware acceleration by processing large volumes of data in parallel, increasing throughput in comparison to a CPU or microcontroller.

In inverse kinematics problem, to increase the end-effector's accuracy and precision, float-point data type should be used, which introduces severe time delay compared to fixed point calculation, also, high rank of matrices like 8*8, 16*16 are needed for redundant robot to express their multiple DOF features, the multiplication steps between internal procedure of getting IK would be thousands of times. The resource and timing demanding feature constraints the localized calculation inside the robot processing system, and portability is also limited due to the need for a powerful computer.

Therefore, for speed up the computational time, the inverse kinematics method based on FPGA realization is studied in this paper. And the C++, as well as high-level synthesis tool, is applied to generate the inverse kinematics hardware

[1]Yike Li is a student of School of Information and Science Technology, ShanghaiTech University, 393 Middle Huaxia Road, Pudong, Shanghai yike.pear@gmail.com

[2,3]Lu Li and Howie Choset are from Robotic Institute of Carnegie Mellon University lilu12@andrew.cmu.edu, choset@andrew.cmu.edu

accelerator core.

II. Methods

A. Inverse Kinematics problem description

A basic idea of solving inverse kinematics, or tracking the target positions, is that the robot should try to reach out the target position when they are distant. Or, mathematically, a robot should try to minimize the error distance between the target position and end-effector position. Also, when considering the cases that the target position is unreachable because of link length is limited or joint angle needs space to rotate, the equation should be:

$$\boldsymbol{e} = f_{ClampMag}(\boldsymbol{t} - \boldsymbol{s}, D_{max}), \tag{1}$$

where

$$f_{ClampMag}(\boldsymbol{w}, d) = \left\{ \begin{array}{ll} \boldsymbol{w} & \text{if } \|\boldsymbol{w}\| \leq d \\ d\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} & \text{otherwise} \end{array} \right\} \tag{2}$$

$\boldsymbol{t}, \boldsymbol{s}$ denote the target position and end-effector position. D_{max} denotes the maximum distance that the end-effector can move within a single step.

Evidence [10] shows that by defining $\boldsymbol{e}$ in this way, the oscillation that happens when target are out of reach can be effectively reduced.

B. The pseudo inverse method

Jacobian Matrix can be used to describe the local change of each joint angle in a single move, because it is the matrix of all first-order partial derivatives of the position vector function. It can be described in the form as

$$\boldsymbol{J} = \begin{bmatrix} \frac{\alpha f_1}{\alpha x_1} & \cdots & \frac{\alpha f_1}{\alpha x_n} \\ \vdots & \ddots & \vdots \\ \frac{\alpha f_m}{\alpha x_1} & \cdots & \frac{\alpha f_m}{\alpha x_n} \end{bmatrix} \tag{3}$$

where $f_1 to f_m$ can be the positions x, y, z and angles roll, pitch, yaw of the end-effector.

The desired angle change of each joints $\Delta\boldsymbol{\theta}$ can be found by equation:

$$J\Delta\boldsymbol{\theta} = \boldsymbol{e} \tag{4}$$

For redundant robotics, m is not equal to n, so the Jacobian matrix is not square or not of full rank. To make sure there exists the solution, left multiply the transpose of Jacobian matrix:

$$J^T J\Delta\boldsymbol{\theta} = J^T\boldsymbol{e} \tag{5}$$

If J has full row rank, then JJ^T is invertible, so the solution of $\Delta\boldsymbol{\theta}$ can be obtained by:

$$\Delta\boldsymbol{\theta} = J^T(JJ^T)^{-1}\boldsymbol{e} \tag{6}$$

A more general formula that for both J that has full row rank or not is given by Buss[11]. Even though, this method still performs poorly because it has a singularity problem. When the configuration is close to a singularity, the pseudo-inverse method will be sensitive to a small target position change, which will introduce big torque to conduct rapid change in joint angle.

C. Levenberg-Marquardt method

This method was first used by Wampler [12] and Nakamura [13], also called as damped least square method. The advantage of this method compared to the pseudo method is that it can avoid the unstable problem of singularity and give a numerically stable solution for $\Delta\boldsymbol{\theta}$. According to Wampler's justification [14], the value of $\Delta\boldsymbol{\theta}$ can minimize equation (7):

$$\|J\Delta\boldsymbol{\theta} - \boldsymbol{e}\|^2 + \lambda^2 \|\Delta\boldsymbol{\theta}\|^2 \tag{7}$$

λ is the damping factor, it should be large enough to avoid collision near singularities, and λ value varies from different robotics configuration and task, plenty of research [15] [16] [17] [18] proposed dynamical methods to select damping factor based on the situation. In this paper, we assume that the damping factor is given by the user, which means this is a known constant.

To minimize equation (7), it is mathematically equivalent to minimizing:

$$\left\| \begin{pmatrix} \mathrm{J} \\ \lambda\mathrm{I} \end{pmatrix} \Delta\boldsymbol{\theta} - \begin{pmatrix} \boldsymbol{e} \\ \mathbf{0} \end{pmatrix} \right\| \tag{8}$$

To ensure the existence of solution, do the left modification:

$$(J^T J + \lambda^2 I)\Delta\boldsymbol{\theta} = J^T\boldsymbol{e} \tag{9}$$

So the solution to $\Delta\boldsymbol{\theta}$ is:

$$\Delta\boldsymbol{\theta} = (J^T J + \lambda^2 I)^{-1} J^T\boldsymbol{e} \tag{10}$$

To further reduce the possible calculation times, the formula $(J^T J + \lambda^2 I)^{-1} J^T$ can be rewritten as $J^T(JJ^T + \lambda^2 I)^{-1}$,

$$\Delta\boldsymbol{\theta} = J^T(JJ^T + \lambda^2 I)^{-1}\boldsymbol{e} \tag{11}$$

So the $n \times n$ matrix $J^T J$ can be converted to $m \times m$ matrix JJ^T, where m usually indicates the six dimension of target space, $(x, y, z, roll, pitch, yaw)^T$, and n indicates the dimension of degree of freedom(DOF), which is often bigger than m in redundant robot.

In this paper, we assume that Jacobian matrix J, damping factor λ and target position $\boldsymbol{t}$ as well as end-effector position $\boldsymbol{s}$ are already given, so the problem will be focused on how to design FPGA configuration to accelerate the computation.

III. Implementation

In this paper, we use $MiniZed^{TM}$ board. It is a single-core Zynq 7Z007S development board. This board provides a low-cost prototyping platform to fast develop and verify customized design for users. The board detail is showed in Fig.1. The chips and parts used in the design are boxed in the red line.

MiniZed Board has external 512MB DDR3 memory, and it is easy to reprogram the configuration as well as debug via on-board USB to JTAG and debug UART circuit. The core 7Z007S integrates a single-core ARM $Cortext^{TM} - A9MPCore^{TM}$ based processing system(PS) and Xilinx

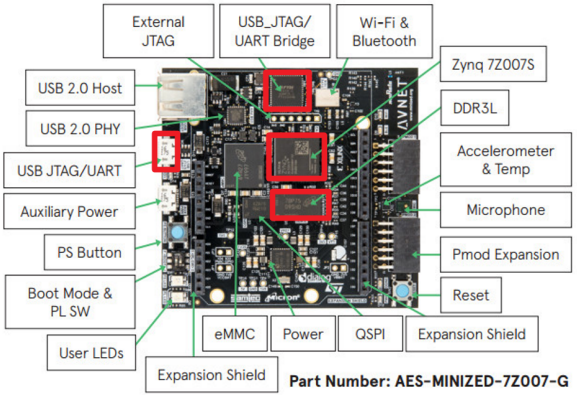


Fig. 1. MiniZed detail view.

programmable logic(PL) in a single device. The development tool we used is Xilinx tool Vivado, software development kit(SDK) and High level synthesis.

A. System design

The goal of the system is to fast calculate high dimensional inverse kinematics problems, therefore, we choose 6, 8, 10 and 16 joints robotic scenarios to test the accelerator performance.

To fulfill the demand of computing in real time, we design the system (Fig.2) with mainly three blocks, the processing system(PS), direct memory access(DMA) and customized programmable logic IP. AXI4-stream is used for high-speed streaming data transformation between different parts.

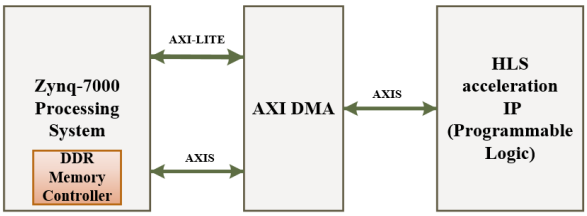


Fig. 2. System design and interconnects between PS and PL

Besides these functional parts, there are also a timer to count acceleration block's calculation time, some interconnects between AXIS slaves and masters, also, a contact and system reset to synchronize the interruption and clock.(Fig. 4)

B. AXI4-stream protocol

Compared to AXI4 and AXI4-Lite, AXI4-Stream get rid of memory-mapped interfaces, therefore removes the requirement for an address phase altogether. It allows unlimited data burst size, therefore, the applications of AXI4-Stream are typically focused on a data-centric and data-flow design, where the concept of an address is not present or not required.

C. Processing system and DMA

The processing system is used to control DMA engine to transfer the data. AXI DMA IP core provides high-bandwidth direct memory access between the AXI4 memory mapped, which is external memory DDR3, and AXI4-Stream IP interfaces. The central processing unit(CPU) inside the ARM core can send commands to initialize, check status and manage registers inside the DMA engine.

D. HLS acceleration IP

This IP is designed to have stream data interfaces as input and output. And the function of this IP is to compute and accelerate the computation from input data, and then send out to the output port. The idea of acceleration is to utilize FPGA's parallel processing feature, so series operations like looping can be unrolled to parallel synchronized operation.

1) Design tool: In this paper, we used the Xilinx tool High-Level Synthesis(HLS) to transforms the C specification, which is C++ code, into a Register Transfer Level(RTL) implementation that synthesizes into FPGA. HLS design can directly compile C++ code and generate the corresponding implementation. It also supports simulation and debugging, which is more user-friendly and efficient for complex project design. The most important thing is that HLS can create many different implementations from the source code using optimization directives which improves the likelihood of finding the most-optimal implementation while sacrificing a little amount of time. There is another way to generate RTL design by using hardware description languages like VHDL or Verilog. However, the latter one is time-consuming because it is a low-level description language that reaches the register level assignment, also, it is difficult to re-customized to different applications.

2) Interface and Acceleration procedure: This block reads data directly from external memory via DMA, then after computation, gives results back to external memory, read and write procedure are controlled by processing system command. The working flow is shown in Fig.3

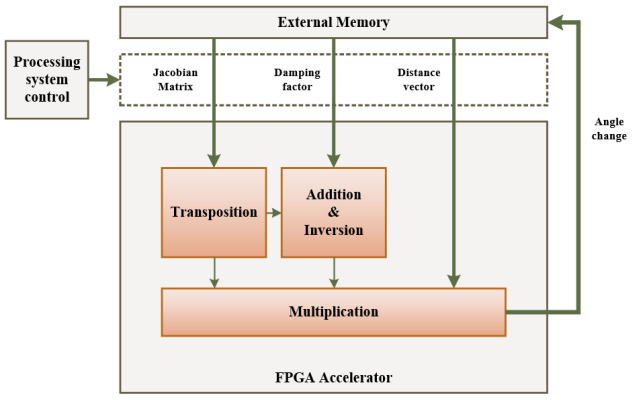


Fig. 3. Working principle of FPGA.

When the accelerator gets the Jacobian matrix, damping factor and distance vector from external memory, it computes the equation (11).

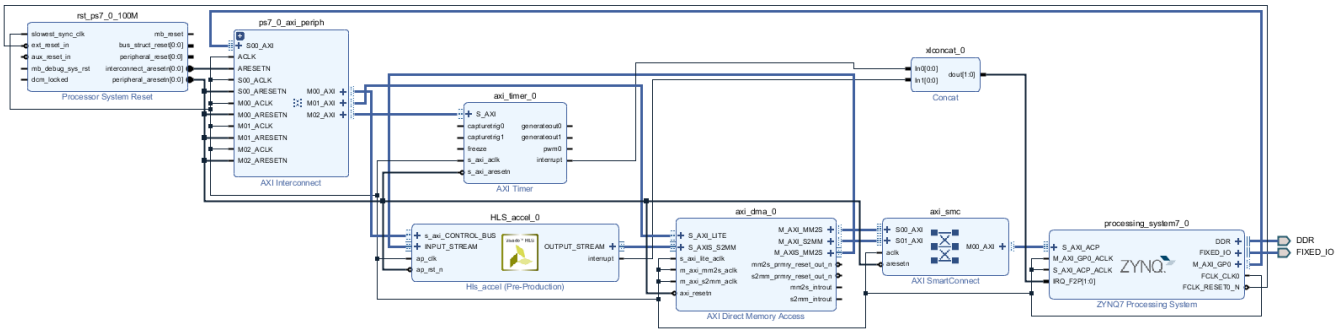


Fig. 4. Vivado block design

3) Matrix inversion: The costly part of the acceleration is getting the inversion of the middle part. We choose the LUP decomposition method to find the matrix inversion. For any given full-rank matrix A, it can be decomposed as a lower-triangle matrix L, a diagonal matrix D, and an upper-triangle matrix U after several possible row permutations. As shown in equation (12).

$$PA = LDU \tag{12}$$

To get the inverse of matrix A, it is easy to know that

$$A^{-1} = U^{-1}D^{-1}L^{-1}P^{-1} \tag{13}$$

U^{-1} and D^{-1} can be obtained by back substitute iterations. Since L and U are triangular matrices,

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ a_{2,1} & 1 & 0 & \dots & 0 \\ a_{3,1} & a_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d,1} & a_{d,2} & a_{d,3} & \dots & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ b_{2,1} & 1 & 0 & \dots & 0 \\ b_{3,1} & b_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{d,1} & b_{d,2} & b_{d,3} & \dots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ a_{2,1}+b_{2,1} & 1 & 0 & \dots & 0 \\ a_{3,1}+a_{3,2}b_{2,1}+b_{3,1} & a_{3,2}+b_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \dots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

So, in the first iteration, all the $b_{n,n-1}$ can be solved, the second iteration, all the $b_{n,n-2}$ can be solved. The iteration ends after all the b elements are found.

D is a diagonal matrix, so the reciprocal is its inverse. P is a row permutation matrix, left multiple P will permute row order, right multiple the inversion of P matrix will permute column in the same order.

All the looping involved in this part are pipelined to reduced waiting time and flatten to be parallel processed.

4) The complete computation: Start with reading data till the end of the computation, FPGA accelerator has to multiply the Jacobian transpose, inverse part and distance vector. In Fig.5, the red parts indicate blocks are accelerated, green parts indicate blocks are not accelerated.

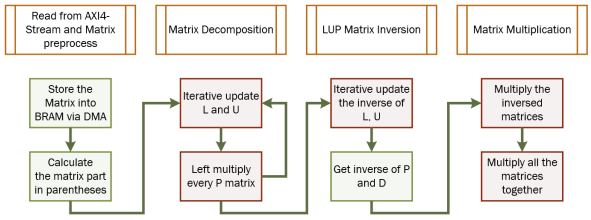


Fig. 5. System design and interconnects between PS and PL

Table 1 shows the resource usage and HLS time performance for different DOF robots

DOF	6	8	10	16
Resource utilization				
LUT	7459	7486	7486	7486
FF	6227	6243	6243	6243
DSP	30	30	30	30
BRAM18K	12	12	12	12
Timing				
Latency(max)	4133	4637	5141	6647
Interval(max)	4132	4636	5140	6646
HLS time(ms)	0.08265	0.09273	0.10281	0.13293

TABLE I
RESOURCE USAGE AND HLS TIME PERFORMANCE FOR DIFFERENT DOF ROBOTS

The results show that even for DOF range from 6 to 16, there are only a few differences in terms of resource usage. However, the execution time increases with the number of robot joints. One reason for this is because current system design that implemented on MiniZed Board uses 85% LUT(look-up table), for resource consideration, the function parts we optimized have to be selectively, so there is a balance between resource and performance.

IV. Experimental Results

We test the implementation on Artix-7 based programmable logic @100MHz and MATLAB on Intel(R) Core(TM) i7-5500U CPU @2.40GHz. Also, since the algorithm involves float data type and inversion operation, we check the data quality by comparing the FPGA result with MATLAB result. We use the square root of mean square error as the metric. Table 2 shows that the powerful software mounted on the powerful CPU only has a speed-up of no more than 10 times than our small chip, also, the error is less than 0.001 for all types redundant robots, which is accurate enough for robots since actuators have limited angular resolution.

	Timing(ms)		**Error**	
DOF	Pure HW	MATLAB	MSE	MAX
6	1.43551	0.579	1.429E-6	1.934E-4
8	1.44057	0.869	4.070E-7	1.696E-5
10	1.44560	0.923	4.217E-7	4.804E-5
16	1.46040	1.014	1.982E-7	1.677E-6

TABLE II

Resource usage and HLS time performance for different DOF robots

V. Conclusion and Discussion

We designed an inverse kinematics core implemented on a Xilinx MiniZed Board with a Zynq core. By using the Levenberg-Marquardt algorithm with 32-bit float-point arithmetic, the 6 DOF, 8 DOF , 10 DOF and 16 DOF robot IK solutions were solved in real time with 100 MHz frequency. The benchmark test is investigated for different joint numbers with different processing devices. The proposed design can achieve a throughput of 696, 694, 691, 684 updates per second for 6, 8, 10, 16 joints robots. The proposed design can reach same order of magnitude timing compared with a strong powerful CPU for 16 DOF redundant robot, while the size and prize are more than 10 times lower than the latter one. Our design can be easily customized to different robots with different configurations and Jacobian matrix sizes.

The major constraint on our implementation is we didn't fully parallel all the progress, the results show that with the number of DOF increase, the resource does not increase, however, the time does, which means the critical time relies on the series part. One possible solution is that we change our device with a board that has enough resource, so we can flatten all progress so that they can run concurrently to reach a better performance. From the resource perspective, we can also conduct a conclusion that current design is well pipelined and might be capable with more DOF since the resource usage for 8, 10, 16 DOF is exactly the same. In the future, several experiments will be conducted to evaluate our proposed system on physical robotic hardware such as industrial robot arms and legged mobile robots, also, we will explore the possibility to play with more complex robots.

ACKNOWLEDGMENT

I would like to thank ShanghaiTech University for this funded research opportunity. I would especially like to thank Dr. Lu Li and Professor Howie Choset for their advising and mentoring, I would also like to thank Rachel Burcin, John Dolan, Ziqi Guo, and the rest of RISS cohort for their support.

References

[1] L. . T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, Aug 1991.

[2] A. C. Nearchou, "Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm," *Mechanism and Machine Theory*, vol. 33, no. 3, pp. 273 – 292, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094114X97000347

[3] X. Wang, D. Zhang, and C. Zhao, "The inverse kinematics of a 7r 6-degree-of-freedom robot with non-spherical wrist," *Advances in Mechanical Engineering*, vol. 9, no. 8, p. 1687814017714985, 2017. [Online]. Available: https://doi.org/10.1177/1687814017714985

[4] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Trans. Graph.*, vol. 13, no. 4, pp. 313–336, Oct. 1994. [Online]. Available: http://doi.acm.org/10.1145/195826.195827

[5] R. Grzeszczuk and D. Terzopoulos, "Automated learning of muscle-actuated locomotion through control abstraction," in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 63–70. [Online]. Available: http://doi.acm.org/10.1145/218380.218411

[6] E. Oyama, N. Y. Chong, A. Agah, and T. Maeda, "Inverse kinematics learning by modular architecture neural networks with performance prediction networks," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 1, May 2001, pp. 1006–1012 vol.1.

[7] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, no. 2, pp. 47–53, June 1969.

[8] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *The 23rd IEEE Conference on Decision and Control*, Dec 1984, pp. 1359–1363.

[9] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, Jan 1986.

[10] 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.

[11] ——, *3D computer graphics: a mathematical introduction with OpenGL*. Cambridge University Press, 2003.

[12] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[13] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of dynamic systems, measurement, and control*, vol. 108, no. 3, pp. 163–171, 1986.

[14] C. Wampler and L. Leifer, "Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, no. 1, pp. 31–38, 1988.

[15] S. K. Chan and P. D. Lawrence, "General inverse kinematics with the error damped pseudoinverse," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. IEEE, 1988, pp. 834–839.

[16] S. Chiaverini, "Estimate of the two smallest singular values of the jacobian matrix: Application to damped least-squares inverse kinematics," *Journal of Robotic Systems*, vol. 10, no. 8, pp. 991–1008, 1993.

[17] R. Mayorga, N. Milano, and A. Wong, “A simple bound for the appropriate pseudoinverse perturbation of robot manipulators,” in *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*. IEEE, 1991, pp. 1485–1488.

[18] R. V. Mayorga, A. K. Wong, and N. Milano, “A fast procedure for manipulator inverse kinematics evaluation and pseudoinverse robustness,” *IEEE Transactions on Systems, Man, and cybernetics*, vol. 22, no. 4, pp. 790–798, 1992.

FG-GMM based Social Behavior Estimation for Autonomous Driving Vehicle in Ramp Merging Control

Yiwei Lyu[1], Chiyu Dong[2],and John M. Dolan[3]

***Abstract*— Social behavior is important for autonomous driving vehicles, especially for scenarios like ramp merging which require significant social interaction between autonomous driving vehicles and human-driven cars. This project enhances our previous Probabilistic Graphical Model (PGM) merging control model for the social behavior of autonomous driving vehicles. To better estimate the social behavior for autonomous driving cars, a Factor Graph (FG) is used to describe the dependency among observations and estimates other cars' intentions. Real trajectories are used to approximate the model instead of human-designed parameters. Forgetting factors and a Gaussian Mixture Model (GMM) are also applied in the intention estimation process for stablization, interpolation and smoothness.**

The advantage of the factor graph is that the relationship between its nodes can be described by self-defined functions, instead of probabilistic relationships as in PGM, giving more flexibility. The proposed method enhances the performance of the previous solution by 16 percent in terms of collision rate.

I. INTRODUCTION

Since autonomous driving vehicles cannot replace all human driven-cars in the near future, they should share roads with human drivers. Therefore, understanding human drivers' intentions is essential for safe autonomous driving.

Ramp merging is one of the most important and typical scenarios where autonomous vehicles interact with human-driven cars. An autonomous vehicle may collide if it can't correctly predict human drivers' intention given a complicated situation. Therefore, it is important for autonomous vehicles to understand human drivers' intentions and generate appropriate social behaviors. Since human-driven cars can introduce significant uncertainty in autonomous-human driving interaction, it is challenging to estimate their intentions.

Current autonomous driving vehicles are not able to understand social behavior properly. Human-driven cars' intentions can not be predicted by simply using the speed detection sensors like RADAR, which will introduce large noise that results in serious estimation oscillation. This task is more challenging if there is no V2X communication system. In our previous research, a probabilistic graphical model (PGM) is proposed to model uncertainty with probabilities. By taking advantage of the independence and conditional independence that hold among random variables, the representational and computational bottlenecks are alleviated.

*This work was supported by NSF REU funding.

[1]Yiwei Lyu is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

[2]Chiyu Dong is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

[3]John M. Dolan is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

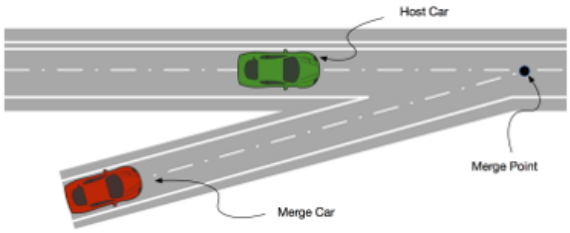


Fig. 1: Ramp merging scenario. The host car (green) is an autonomous vehicle, running on the main road; the merge car (red) is a human driven car, running on the ramp.

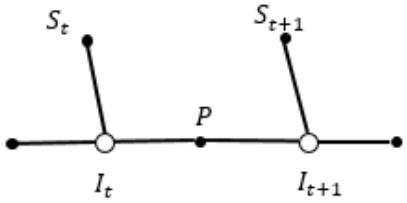


Fig. 2: A factor graph representation for the ramp merging intention estimation. S_t and S_{t+1} are factor nodes which take velocity observation, and time-stamps; I_t and I_{t+1} are latent variables which denote intentions over time; P is a prior factor, which denotes the effect of prior intention.

However, one of the major problems of PGM is that the relationship between nodes in PGM has to be probabilistic, i.e., all transition models should integrate to one. In addition, PGM cannot treat historical data with differing weights.

In this paper, a method based on Factor Graph [1], [2] is proposed. Our main contributions are: 1) using factor graph to better explain the relationship between variables. 2) using a forgetting function to assign self-defined factors to historical data and a Gaussian Mixture Model (GMM) to better approximate the intention prediction.

II. RELATED WORK

There are several works that address the merging problem. Adaptive Cruise Control (ACC) is one of the approaches which is being widely commercialized. For example, GM's Full Speed Range ACC and Audi's "STOP and GO" ACC makes it possible for cars to follow other cars in dense traffic with low speed. ACC is also used by Mercedes-Benz in its lane departure prevention system and Tesla in "Autopilot" to

perform higher-speed autonomous driving. However, these methods cannot help vehicles generate social behaviors to interact with other cars properly.

A slot-based approach is proposed by Marinescu et al. [3], for cooperative intelligent vehicles in on-ramp traffic merging. Its contribution includes preventing congestion under heavy traffic conditions by predicting the time to arrival to the merging point accurately. However, their decision is based on current states and no historical data are considered, which can lead to failures in some cases.

Wei et al. [4] used a hard-coded distribution and cost function to estimate merging intention. They proposed an intention-integrated framework to enable an autonomous car to perform cooperative social behavior, but only instantaneous acceleration is considered. The lack of historical data leads to instability in estimated intention, which results in oscillation or delayed reaction to the autonomous vehicle.

Dong et al. [5] proposed a PGM learning-based method for intention estimation. The main contribution is that the model is trained and validated by real trajectories, and shows great improvement when validated by a designed merging strategy in simulation compared with previous methods. However, the model only takes speed and time as factors in the process of intention estimation. In addition, the transition models trained from data still rely on the Markov assumption.

We use a factor graph which is converted from the original probabilistic graphical model (PGM) to describe dependency among observed data and estimate other car intentions. Real driving data are used to parametrize this model instead of manually designed parameters.

The social behavior estimation can be extended to various cooperation situations, such as lane changing, stop sign traversal and ramp merging. In this paper, we focus on ramp merging.

III. Factor Graph-Based Intention Estimation

A. Structure of Factor Graph

Factor graph is widely used in perception [6] and motion planning [7]. We here adapt it to behavioral estimation. A factor graph representation for ramp merging intention estimation is shown in Fig. 2.

Our model assumes that human intention does not oscillate as fast as the program's update rate. Therefore, one intention node will affect the next n speed nodes. These n speed nodes keep track of the target vehicle's speed during n cycles. The past state (speed) and last intention will decide the intention.

$$P(I_{t+1}|I_t, V_{t-n}, ..., V_t) \tag{1}$$

where I_t is the last intention, and V_{t-n}, ..., V_t are the past $n+1$ velocities. Alternatively, Equation 1 can be written as,

$$I_{t+1} = \arg\max_{I_{t+1}} f(I_{t+1}, I_t, V_{t-n}, ..., V_t) \tag{2}$$

where $f(I_{t+1}, I_t, V_{t-n}, ..., V_t)$ is a factorization of current intention, last intention, and past $n+1$ velocities.

There are two kinds of effects on future intention estimation from other nodes/edges in the factor graph.

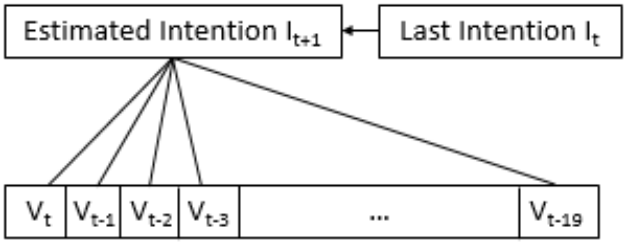


Fig. 3: States for velocities in the factor graph.

1) State (speeds) Effect: The state effect is indicated in Equation 3, which takes historical speed data into consideration during the future intention estimation process. Forgetting Functions $\omega(i)$ are included in the factor nodes and assigned to historical speed data. As shown in Fig. 3, each frame of selected historical speed data will affect the current intention estimation. The forgetting function makes it possible that the closer the historical speed is to the current one, the larger effect it will have on current intention estimation.

$$g(S_{t+1}) = \prod_{i=1}^{n} \omega(i) f(V_i, V_{i-1}) \tag{3}$$

where $f(V_i, V_{i-1})$ describes the transitional relationship between current velocity and last velocity. Here $(V_n, ..., V_i, V_0)$ correspond to the last n+1 frames of past velocity $(V_t, V_{t-1}, ..., V_{t-n})$. The transitional relationship for each current-past velocity pair is gained from the Speed Transition Model, which is introduced in a previous paper [5].

2) Last Intention Effect: This is the effect of the prior intention's effect on future intention estimation, which is indicated in Equation 4. The $b(I_t)$ and $b(I_{t+1})$ in Equation 4 are a "Blurring function", which makes the discrete intention a continuous value ranging from 0 to 1. The Blurring functions help to smooth the estimated intention and encourage the stability of intentions over time.

$$m(I_t) = \exp\{-\frac{||b(I_{t+1}) - b(I_t))||^2}{\sigma}\} \tag{4}$$

where σ is obtained from a Gaussian Mixture Model, which is introduced in Section IV-A.

The Factorization F = $f(I_{t+1}, I_t, V_{t-n}, ..., V_t)$ is expressed as:

$$\begin{aligned} F &= g(S_{t+1}) \cdot m(I_t) \\ &= \prod_{i=1}^{n} \omega(i) f(V_i, V_{i-1}) \cdot m(I_t) \\ &= \prod_{i=1}^{n} \omega(i) \exp(-\frac{||\Delta V(I_{t+1})||^2}{\sigma}) \cdot \exp(-\frac{||\Delta b(I_{t+1})||^2}{\sigma}) \end{aligned} \tag{5}$$

where $\Delta V(I_{t+1}) \triangleq V(I_{t+1}) - V(I_t)$, and $\Delta b(I_{t+1}) \triangleq b(I_{t+1}) - b(I_t)$.

B. Evaluation of Factor Graph

For the forgetting functions in the State effect, we define exponential functions to assign different weights to historical data. This has two main advantages:

1) Prevent Underflow: With exponential forgetting functions, the original product expression is converted to a summation. In the product expression, with tiny values, it's easy to have an underflow situation. Using exponential terms instead, every term will be added up together, which can be efficiently calculated and prevent the underflow situation.

2) Reduce Computation: Converting the original product expression to a summation with exponential terms requires less computation (summation vs. multiplication).

C. Intention Estimation Procedure

The last 20 frames of historical speed data before the merging point $(V_t, V_{t-1}, ..., V_{t-19})$ are selected to be taken into state effect consideration. Forgetting decay functions are assigned to each frame's historical speed datum. The estimated intention I_{t+1} is affected by the 20 frames of historical data, last intention I_t, and the prior term P which assigns weight to the last intention effect.

If the merging vehicle's estimated intention is 'Not Yield', it will tend to speed up and reach the merging point before the host vehicle. Then the host vehicle activates a distance-keeping model to keep a desired safe longitudinal distance to the merging car on the ramp. On the other hand, if its estimated intention is 'Yield', the host car will ignore the merging car and accelerate to the speed limit.

IV. Training from Data

In the previous research [4], prediction of the merging cars behavior was based on hard-coded cost functions and assumptions about the probability distribution of acceleration.

We instead use the US-101 and I80 freeway real-world dataset NGSIM to extract a model of cooperative behavior between host and merging vehicles. The dataset was obtained from overhead cameras near the US-101 and I80's entrance ramps in the Los Angeles area. Cars in this region were filmed and tracked during morning rush hours (7:50 am to 8:35 am for US101, 4:00 am to 5:30 am for I80). The road segment consists of 5 lanes and one entrance ramp at the beginning, as shown in Fig. 4 [8].

Vehicles in the right-most lane on the main road are considered host vehicles, and counterparts on the entrance ramp are considered merging vehicles. We preprocessed the data to filter out unrelated cars that run in inner lanes without interacting with merging vehicles, and used only those from the right-most lane and the entrance ramp.

Host vehicles are paired with merging vehicles that are close to and temporally overlapped with the host, as shown in Fig. 5. There were 647 host-merging vehicle pairs in the dataset. We classify merging vehicles into two classes: 1) yield; 2) not yield, based on which car reaches the merging point first. From group 1, the distribution of $P(V|I = \mathbf{Y})$ can be estimated; from group 2, the distribution of $P(V|I = \mathbf{N})$.

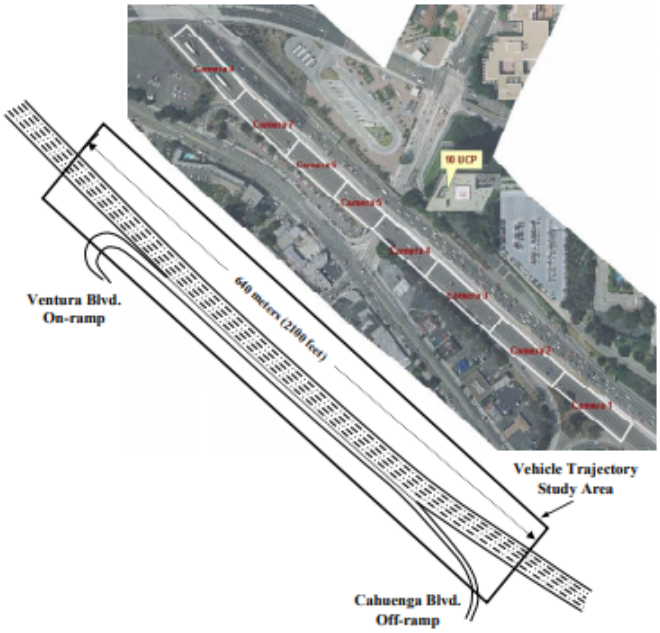


Fig. 4: The aerial photograph above shows the extent of the US 101 study area in relation to the building from which the digital video cameras were mounted and the coverage area for each of the eight cameras. The schematic drawing [8] on the bottom shows the number of lanes and location of the on-ramp at Ventura Boulevard and the off-ramp at Cahuenga Boulevard within the US 101 study area.

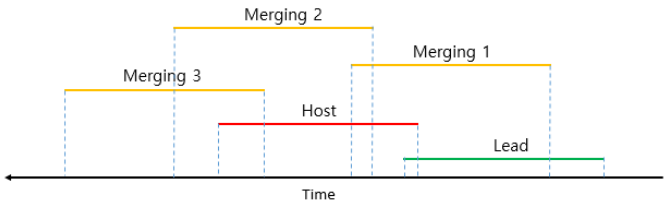


Fig. 5: Time overlap example for a host-leading pair, and multiple host-merging pairs. Three different colors, green, red and yellow indicates three classes of vehicles, the lead vehicle, the host vehicle, and merging vehicles. The length of the colorful lines indicates their corresponding start time and end time for appearance in the camera.

A. Speed Transition Model

The goal of the training is to estimate the conditional probability of intentions given historical speed information, i.e., $P(V|I)$. The two classes of data are used to train the speed transition model. Fig. 6 and Fig. 7 show the speed transition probability distributions for given speed and intention.

The x-axis V_t indicates the current speed of merging vehicles, the y-axis V_{t-1} indicates the past speed of merging vehicles, and the z-axis shows the probability of a particular speed transition occurring. The main difference between the two figures is in the zone 80 (feet/second) $\leq V(t) \leq$ 100 (feet/second) and 80 (feet/second) $\leq V(t-1) \leq 100$ (feet/second), where there are more dots when the intention is 'Not Yield'. If the merging car's driver decides not to yield to the host, it has higher probability to accelerate; and

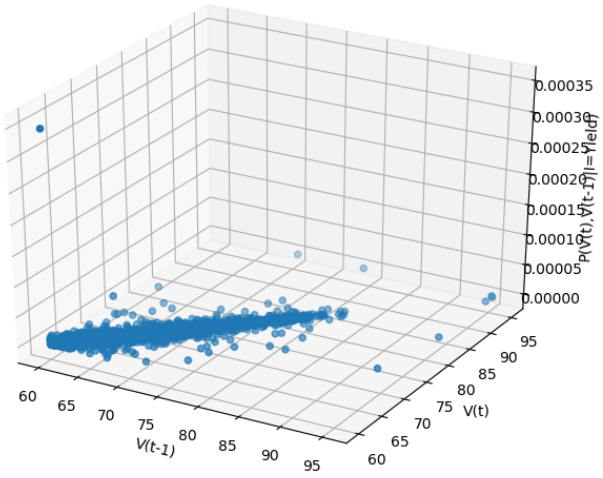


Fig. 6: Speed transition distribution for given speed (feet/second) and Yield intention.

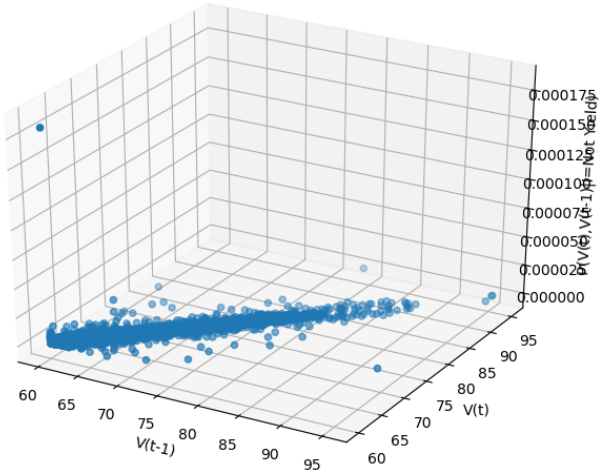


Fig. 7: Speed transition distribution for given speed (feet/second) and Not Yield intention

the merge car is more likely to decelerate if it decides to yield to the host. These results are consistent with intuition. Additionally, each speed has specific transition probabilities under different intentions.

B. Gaussian Mixture Model (GMM)

The speed transition model is also used in the previous PGM-based method. However, there are several shortcomings of the speed transition model:

1) Inefficient Computation Process: In the speed transition model, given current and past speed, a closest coordinate is searched in the two spaces and its probability of occurrence is taken for comparison between 'Yield' or 'Not Yield'. The searching process takes time and is inefficient.

2) Missing Data: In some cases, an existing point close to the target coordinate may not be easily found. In the speed transition model, surrounding points' average probability of occurrence are calculated. This may also lead to inaccuracy. To tackle the problems above, a Gaussian Mixture Model [9] is used to approximate the speed transition probability distribution. A 2-component GMM models is built, as shown in TABLE I.

We assume the distribution of speed transition as a mixture of multiple single Gaussian models. Means μ, covariance σ, and weights ω gotten from the GMM are used to calculate the probability for yield and not yield.

$$P(I) = \sum_i A_i \exp \frac{||x - \mu_i||^2}{\sigma_i}, \tag{6}$$

where

$$A_i = \frac{\omega_i}{\sqrt{2\pi}\sigma_i} \tag{7}$$

V. Experimental Results

We run simulations on reacting to merging veihicles with real-data trajectories which are extracted from datasets. The host car will estimate the merging vehicles' intention and make the decision by observing their state. Simulations were run as experiments on the proposed model. 100 host-merging pairs extracted from the I80 dataset with time range from 05:15 am to 05:30 am were used for testing. We conducted three sets of experiments: 1) Intention estimation without forgetting factor and GMM; 2) Intention estimation with forgetting factor but without GMM; 3) Intention estimation with both forgetting factor and GMM. Collision Rate is calculated as the rate of total collided pair counts among all tested pairs. The experimental results are shown in TABLE II.

From TABLE 2, we find that the collision rate without forgetting factor and GMM is higher than the one only without GMM. The forgetting factor reduces the collision rate by 9 percent. More importantly, the collision rate without GMM is twice that with GMM, which indicates that the GMM greatly improves the accuracy of estimated intention.

VI. Conclusion

Real data test results show that the proposed Factor Graph-based method with Forgetting function and GMM has the lowest collision rate.

Compared with the previous PGM-based method, the collision rate is not lower, probably because the parameters of the model have not been fine-tuned and the forgetting function has not been better designed yet.

In the future, we will further tune the parameters and design the forgetting function. More tests will be run on both US101, I80 and other datasets. The results of the proposed method will be further compared with previous methods, e.g. ACC, Slot, i-PCB, and PGM. Criteria including minimum safe distance, cycle rate, and other terms will be applied.

We will also extend our method to estimate long-term motion of merging vehicles, in the scenarios that have more than one merging car, and take advantage of a broader set of training data.

TABLE I: 2-components Gaussian Mixture Model parameters

Yield		Not Yield	
Means	[[17.59 17.58][43.51 43.50]]	Means	[[44.02 44.01][17.87 17.87]]
Covariance	[[64.04 64.00][106.53 106.77]]	Covariance	[[106.96 107.20][66.25 66.20]]
Weights	[0.57 0.42]	Weights	[0.43 0.56]

TABLE II: Collision Rate Comparison

Dataset	Num. of Pairs	Collision Rate(w/o FF,GMM)	Collision Rate(w/o GMM)	Collision Rate(w/ GMM)
I-80 (05:15-05:30)	100	23	14	7

ACKNOWLEDGMENTS

I would like to thank Dr. John Dolan and Mr. Chiyu Dong for their advising and mentoring. I would also like to thank the RISS program for this funded research opportunity and the entire RISS team and cohort for help and support.

REFERENCES

[1] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004.

[2] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[3] D. Marinescu, J. Čurn, M. Bouroche, and V. Cahill, "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 900–906.

[4] J. Wei, J. Dolan, and B. Litkouhi, "A prediction-and cost function-based algorithm for robust autonomous freeway driving," in *Intelligent Vehicles Symposium*, 2010.

[5] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *2017 IEEE 28th Intelligent Vehicles Symposium (IV'17)*, Jun. 2017, pp. 1584 – 1589.

[6] F. Dellaert, M. Kaess *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[7] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs." in *Robotics: Science and Systems*, vol. 12, 2016.

[8] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, "The next generation simulation program," *Institute of Transportation Engineers. ITE Journal*, vol. 74, no. 8, p. 22, 2004.

[9] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

What's Most Broken? A Tool to Assist Data-Driven Iterative Improvement of an Intelligent Tutoring System

Shiven Mian*[1] Mononito Goswami*[2] Jack Mostow[3]
Indraprastha Institute of Information Technology, Delhi, India[1]
Delhi Technological University, Delhi, India[2]
Carnegie Mellon University, PA, USA[3]
shiven15094@iiitd.ac.in[1], mononitog@hotmail.com[2], mostow@cs.cmu.edu[3]

***Abstract*— Intelligent Tutoring Systems (ITS) have great potential to change the educational landscape by bringing scientifically tested one-to-one tutoring to remote and under-served areas. But effective ITSs are too complex to make perfect. Instead, a practical guiding principle for ITS development and improvement is to fix what's most broken. In this paper we present SPOT (Statistical Probe of Tutoring), a tool that mines data logged by an Intelligent Tutoring System to identify the "hot spots" most detrimental to its efficiency and effectiveness in terms of its software reliability, usability, task difficulty, student engagement, and other criteria. SPOT uses heuristics and machine learning to discover, characterize, and prioritize such hot spots in order to focus Intelligent Tutoring System refinement on what matters most. We applied SPOT to data logged by RoboTutor, a Finalist in the $15M Global Learning XPRIZE Competition to help children acquire basic literacy and numeracy without adult assistance.**

***Index Terms*— Intelligent Tutoring Systems, Educational Data Mining, Software Reliability, Student Engagement, Usability, Decision Trees**

I. INTRODUCTION

An Intelligent Tutoring System (ITS) is a computer system that enables learning in an effective and meaningful manner by providing personalized instruction to learners. Intelligent Tutoring Systems are becoming increasingly popular for education across a wide variety of subjects from algebra and geometry to foreign languages. In the 1970s researchers envisioned an ambitious goal for computer-based instruction: inspired by human teachers, they began applying principles of machine learning and cognitive psychology to lend intelligence to these tutors.

Ever since, the field has progressed from simple cognitive tutors to more and more complex tutoring systems that provide customized instruction and feedback to students [21]. Prior research has demonstrated that ITSs take extensive time to author, with reported estimated of 200-300 hours of development per hour of instruction [1]. As a result, many authoring tools such as the Cognitive Tutor Authoring Tool (CTAT) have been built to make ITS development more efficient for programmers as well as non-programmers. Besides, despite our growing understanding of human cognition, the tutor authoring process, and advances in Machine Learning, developing effective tutoring systems remains hard. Koedinger et al [20] identified 30 instructional decisions such as spacing of practice, concreteness of examples, timing of feedback etc., for authors of tutoring systems.

As our own Intelligent Tutoring system, we present RoboTutor [28] (see Figure 1): an ITS developed as an open-source Android application that teaches children aged 7-10 in developing countries basic reading, writing, and arithmetic. As one of the five finalists in the Global Learning XPRIZE [14] competition, RoboTutor was designed based in part on lessons from over two decades of educational research on Project LISTENs Reading Tutor [27].

But with limited development resources and time, Intelligent Tutoring Systems become very hard to make perfect. Instead, a practical guiding principle is 'fix whats most broken'.

Figure 2 illustrates the iterative development process of RoboTutor. The process involves designing prototype activities, deploying them for field testing, collecting data as the children do the activity, and mining this data to identify and modify "whats most broken."

In this paper, we address the question: Is there a way to use data from RoboTutor to automate discovery of design issues and highlight them? To best address this issue, we must answer the following subquestions pertaining to the development and design process of RoboTutor:

- **Reliability:** How often and under what conditions does RoboTutor crash or hang? How fast does a child recover from the crash or hang?
- **Recognition**: How accurately does RoboTutor recognize both written and spoken written input?
- **Usability:** How easily and efficiently can children operate RoboTutor? Which activities do they find hard to navigate?
- **Engagement:** When are the children disengaged the most, and why?

To answer these questions, we present Statistical Probe of Tutoring (SPOT), an educational data mining tool intended to help ITS developers identify what's most broken that is, hot spots with respect to design criteria such as software reliability, recognition accuracy, UI/UX usability, student engagement, task difficulty, and learning. SPOT uses quantitative metrics to evaluate such criteria; for example,

*both authors contributed equally

metrics of reliability include the frequency of crashes and hangs.

SPOT uses metrics to identify and predict the occurrence of undesirable events, and trains a decision tree to discover hot spots. A hot spot is a subtree with a high proportion of undesirable events.

The intuitions that inspired this approach are as follows:

- Within a decision tree, undesirable events in the same subtree are likely to have the same underlying cause.
- The feature combination associated with a subtree that is, the sequence of tests from the root to the subtree characterizes when the undesirable events tend to occur and may reflect this underlying cause.
- Screen capture videos of a random sample of undesirable events in the subtree may shed further light on a certain cause and inspire ideas for how to address it.

SPOT may be especially useful when user-testing in person is impractical, such as in situations where the users are far away, when children may behave differently when adults are present, and when hot spots are important to fix but too rare to observe in person.

A senior capstone project in Carnegie Mellons Bachelors of Human-Computer Interaction program designed a UI/UX for SPOT, and prototyped it using simulated data [17]. Here we describe an implementation that uses automated decision tree learning and runs on real data from RoboTutor.

Section II discusses related works. Section III describes SPOT. Section IV summarizes our contributions to Intelligent Tutoring Systems and presents avenues of future work.

II. Relation to Prior Work

Many Intelligent Tutoring Systems record student interactions in log files. These log files are major sources of data for analyzing reliability, usability, student engagement, motivation and learning.

A. Software Reliability

Intelligent Tutoring Systems, like other software products, must be reliable. Frequent crashes or hangs might potentially disengage learners and negatively impact user experience. Prior studies focus on software reliability growth models based on assumptions about the nature of faults and the non-deterministic behavior of failures [15]. Connectionist models have also been used to predict software reliability, as they can generate models automatically from the history of past failures [19]. In our work, since RoboTutor does not explicitly log crashes, we could not use reliability models that utilized data of past failures. Instead, SPOT uses simple heuristics to label likely instances of a crash or hang, eg. session changes on the same tablet with a short hiatus.

B. Usability

Usability is concerned with making a system easy to learn and use. A natural approach to realizing usable ITSs involves iterating between design and usability evaluation until a satisfactory usable design is achieved. Granic et al [16] present a scenario-based usability evaluation approach to ITS evaluation. They evaluated the user interface of an ITS based on objective performance measures of effectiveness and efficiency, and users' subjective assessment of their system use. Their operational definition of usability enables them to set quantitative goals of execution before evaluation, as well as specify operationally defined criteria for success, based on measurable attributes, such as error rate. Furthermore, subjective assessment of usability and user experience, in addition to scenario-based user testing, has often been utilized to assess the usability of ITS. [7], [8], [16].

C. Engagement, Off-task Behavior and Gaming the System

Children must focus on solving problems and engage with a tutor to effectively utilize their time and learn. A child is disengaged when he or she is not actively thinking about the subject material, and/or attempts to "game the system" by systematically misusing the ITSs feedback to advance through the curriculum. Studies such as [9] have shown that off-task, and gaming behaviors are associated with poor learning outcomes . There is also extensive literature on detecting instances of student disengagement and lack of motivation. Beck modeled student engagement based on modified item response theory using a technique called Engagement Tracing [18]. Engagement tracing modeled disengagement without humans having to rate user interactions or measurements with biological sensors, and showed that time on task is an important predictor of disengagement.

Some years later, Cocea et al [10] used a richer bag of features from log files to train a decision tree and detect students motivation levels. These studies concluded that student engagement can be effectively determined from basic data recorded in log files.

Learner disengagement often manifests itself as gaming behavior in which students take undue advantage of the tutors feedback and hint architecture. Baker et al [3], [4], [12] extensively examined gaming behavior and developed latent response models to classify instances of gaming the system.

Successful detectors of engagement and off-task behavior in the past have often relied on either a limited set of activities (or questions), for example [18] examined student performance on multiple-choice cloze questions only, or field observations of student affect to train statistical or machine learning models [9]. In our work, we did not have any data from observers to ascertain whether a child is off-task while using RoboTutor. Also, RoboTutor has many different kind of activities such as story reading activities, multiple choice activities, and writing activities.

Metrics such as number of bailouts in activities, and average duration per attempt (see Table II) under Engagement were inspired by predictors of gaming behavior. Studies [5], [10], [11], [13], [18] have shown that these predictors are accurate in detecting disengagement.

D. Learning

While tools like [22] that explore logs of student and tutor interaction exist, to the best of our knowledge SPOT is the first tool that specifically facilitates the discovery of

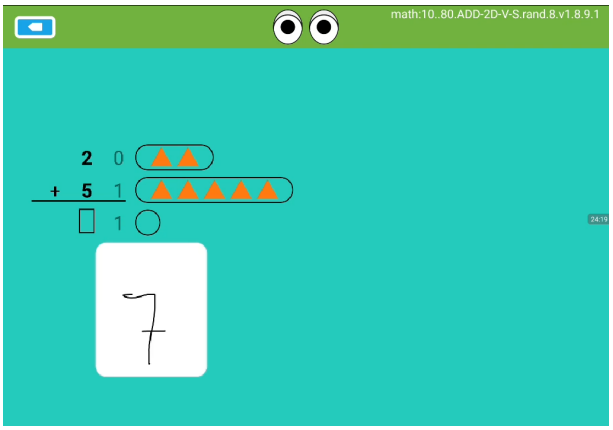


Fig. 1: A math activity in RoboTutor: The child has to evaluate 5 + 2 and write the answer

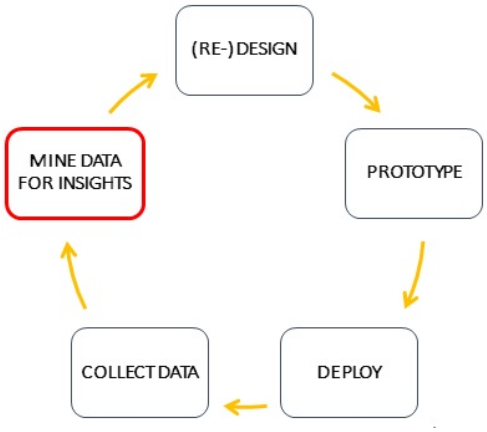


Fig. 2: Data-driven Iterative Development process

design issues. Besides, SPOT relies only on log files and screen captured videos, both of which are easy to record. In contrast, previous studies have utilized field observations and questionnaire evaluations that are often time and resource intensive.

III. METHODOLOGY

A. *Organization of RoboTutor*

RoboTutor uses a variety of activities such as story reading, math activities etc. to teach children basic reading, writing, and math skills. Figure 1 illustrates a math activity teaching addition of two numbers. RoboTutor is organized as follows:

1) **Activity:** A tutor or game that teaches a certain skill, such as addition of two numbers by presenting a sequence of items that exercise the skill.
2) **Item:** An item is a stimulus presented visually and/or orally. Depending on the activity, the child taps, speaks or writes a response. In figure 1, the problem of adding 20 and 51, and writing their sum is an item of a math activity. The next item is shown when the child gets the current item right. An item may require multiple steps.
3) **Step:** A part of an item the child does. For example figure 1 is a two-step item. Adding numbers in the ones' place is the first step, while adding numbers in the units' place is the second step. A step may require many attempts.
4) **Attempt:** A single input by the child within a step. An attempt can be either correct or incorrect. Steps can be composed of more than one attempt, as children are prompted to re-attempt after incorrect attempts. Activities vary in the number of incorrect attempts allowed before RoboTutor automatically advances to the next step.

The RoboTutor log data consists of transactions. A transaction is a record of an attempt at a step of an item, which can be a successful, unsuccessful, or unfinished attempt. An attempt where a child quits an activity is regarded as an unfinished attempt.

We also derived three levels of aggregates from the dataset:

- **Step level:** Aggregation over all the attempts at a step, e.g. number of attempts
- **Item level:** Aggregation over all steps of an item, e.g. time spent on items
- **Activity level:** Aggregation over all items of an activity, e.g. percent of items correct

B. *Data Collection*

The dataset used by SPOT to discover hot spots is derived from the performance logs from RoboTutors beta field testing sites in Tanzania between April and July 2018, and is composed of 357,115 student transactions from a total of 198 user IDs (students aged 7-10), spanning across approximately 212 student hours. After data cleaning, we get the fields in the logs relevant to SPOT, as listed in Table I. SPOT also uses screen capture videos of RoboTutor recorded using AZ Screen Recorder [24], used by developers to further examine the obtained hot spots.

C. *Approach*

SPOT aims to help ITS developers by focusing their attention on design issues that hamper effective tutoring. SPOT automates the discovery of these design issues based on different criteria and quantitative metrics. Figure 3 illustrates the SPOT work flow.

A design criterion can be formally defined as any property of an Intelligent Tutor that we wish to analyze. We classify Design issues into one of these design criteria: Reliability, Recognition, Usability, Engagement and Learning. For example, Figure 5 illustrates a design issue under Recognition, where the writing recognizer tends to mis-classify 7 as 1 in the absence of its middle stick.

A metric is a heuristic that facilitates the search for design issues under a design criterion. For example, the rejection rate of responses in writing activities is a metric of writing recognition. A metric of reliability is: "Small hiatus between session changes of the same child on the same tablet". A

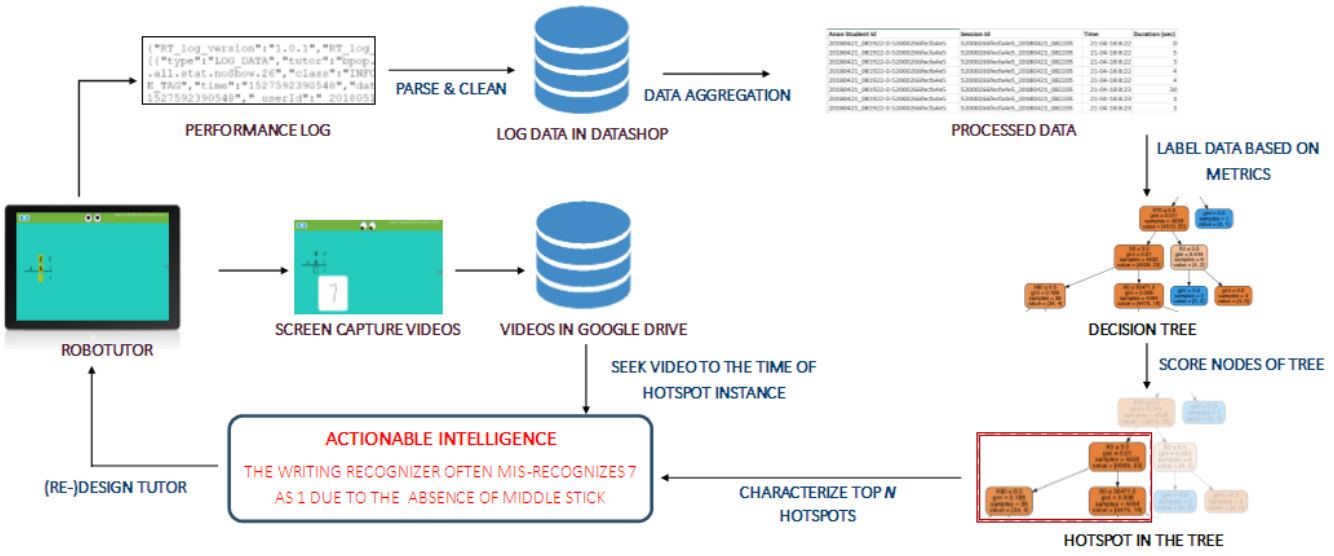


Fig. 3: Workflow of SPOT

short time interval between two sessions of the same child may indicate that RoboTutor crashed, and the child had to restart RoboTutor. Table II enlists various metrics under different criteria.

A metric approximately labels each row of the data (representing either an attempt, item or activity depending on the aggregation) as: a suspicious (positive) instance indicating a design issue, or a non-suspicious (negative) instance. For example, all attempts at writing activities where the responses were rejected, were labeled as suspicious instances. Our labeling qualifies as 'approximate' because a metric does not guarantee finding a design issue. In our example, suspicious instances may also include incorrect responses, besides correct responses that are rejected due to mis-recognition. Only correct responses that are rejected due to mis-recognition are potential design issues with the writing recognizer.

Decision trees have been extensively used in the past for summarizing, generalizing as well as classifying data [2], [23]. Decision trees also enjoy widespread use in educational data mining due to their comprehensibility [25]. SPOT uses the approximately labeled data to train a decision tree for each metric, using features such as the name of activity, student response, expected answer, duration of an attempt etc. Table 1 lists all the features used to train the decision trees, and Figure 3 illustrates a portion of the decision tree for the rejection rate metric. SPOT trains these decision trees using the Classification and regression trees (CART) algorithm. CART is a non-parametric decision tree learning technique that can learn either a classification or regression tree, depending on the type of the dependent variable. CART constructs a decision tree top-down, by choosing a feature that best splits the set of data items at each step.

SPOT uses the decision trees to search for non-random homogeneity in suspicious instances. Our intuition suggests that undesirable events in the same sub-tree are likely to have the same underlying cause. Therefore, a big cluster of suspicious instances might suggest an underlying design issue that needs immediate fixing. A hot spot is a cluster of instances (an instance can be an attempt, step, item or activity) having bad values of a metric for any given criterion. In a decision tree, a hot spot is a subtree that contains a significant number of bad instances. In our example, most suspicious instances having 7 as the expected answer are grouped in the same subtree, and therefore this subtree is a hot spot.

In any decision tree there are as many as 2^n subtrees, where n is the number of nodes in the tree. But every subtree is not a hot spot. For a subtree to be counted as a hot spot it must be pure, i.e. it must concentrate the suspicious instances as much as possible, and it should also contain a significant number of suspicious instances. A subtree must balance both the properties to qualify as a hot spot. Therefore, leaves of a decision tree are not hot spots in spite of being pure, because they do not contain a significant number of suspicious instances. Similarly, the entire tree is not a hot spot in despite covering all the suspicious instances, because it is significantly impure. SPOT searches for hot spots in the trained decision tree using a scoring function. This function evaluates the "heat" of a subtree. Formally, the heat or score of a subtree is a function of how many of its children are bad instances, and how many of the total bad instances lie in the subtree. We operationalize the notion of heat as the F_1 score of a subtree, defined as:

$$F_1 = \frac{2PC}{P + C}$$

where:

$$P(precision) = \frac{\text{number of suspicious instances in hot spot}}{\text{total number of instances in hot spot}}$$

Granularity	Field Type	Fields
Attempt	Feature	Duration (sec) Tutor Name Tutor Level Problem Name Student Input Activity Status Expected Answer Hiatus Tutor Matrix (Category) Outcome Student Repeated Activity (binary) Student Took Hint (binary) Tablet Name Attempt Number
	Other	Session ID Student ID Time of attempt Tutor Sequence Session Tutor Sequence User Village
Activity / Item	Feature	Tutor Name Tutor Level Tutor Matrix (Category) Tutor Sequence Session (for Activity) Tutor Sequence User (for Activity) Problem Name (for Item) # of attempts # of items (for Activity) Activity / Item Duration (sec) Avg, min, max duration of attempt # of correct attempts % of correct attempts # of bail-outs (activity quit) # of scaffolds (i.e hints taken) # of re-attempts # of activity repeats Activity / Item Hiatus
	Other	Time of Activity / Item Start Time of Activity / Item End

TABLE I: Data from Performance Logs

$$C(coverage) = \frac{\text{number of suspicious instances in hot spot}}{\text{total number of suspicious instances in data}}$$

The higher this F_1 score of a subtree, the hotter it is. We term the top N hottest subtrees as the hot spots. Figure 4 illustrates a particular hot spot in a portion of decision tree.

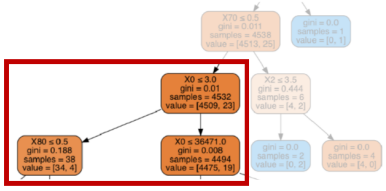


Fig. 4: Hot spot illustration. Nodes in decision tree with high F_1 scores are taken as hot spots

SPOT automatically characterizes each of the hot spots by conjoining every test on the path from the root node of the tree to the hot spot. Hot spot characterization refers to the feature combination associated with a hot spot, and indicates when the undesirable events tend to occur and may reflect this underlying cause. In our example, the hot spot is characterized as: "expected answer = 7."

After discovering a hot spot, SPOT picks up a random sample of suspicious instances, and presents their corresponding screen capture videos to the developers. These screen capture videos may be able to explain the underlying cause or the design issue, and inspire ideas on how to address it. Figure 5 illustrates a screen shot from the screen capture videos for the hot spot characterized as: "expected answer = 7." Looking at the sample of screen capture videos from this hot spot, we realized that the writing recognizer often mis-recognizes 7 in the absence of its middle stick. On examining further videos, we also realised that 7 is most often mis-recognised as 1 in absence of the middle stick, leading children to confuse between the two. SPOT provides links to the exact second by referring to a video metadata table. The video metadata table is a mapping between the performance logs, tablet IDs, video names (file IDs) and location of videos. Since screen record videos of RoboTutor were stored in Google drive, we could design SPOT to display URLs indexed to the exact occurrence of the attempt or item.

Criterion	SPOT Metrics	Positive Labeling Criteria
Reliability	Crashes (quick session changes)	Session ID changes in successive attempts on the same tablet within a very short period (indicative of a crash), mark the first session change attempt as positive
Recognition	Rejection rate	Rejection (Outcome = INCORRECT) for spoken and written input
Usability	Hiatus: time between end of previous attempt / item / activity, and start of the current one	Above mean hiatus
Engagement	# of bailouts in activities	Activities with at least one bailout
	% correct attempts in activities	Activities with % correct < 0.5 and # of incorrect attempts ≥ 2
	Average Duration per attempt	Above mean average duration
	Number of reattempts per item	Number of reattempts ≥ 2

TABLE II: SPOT Metrics and Labeling Criteria

IV. RESULTS, CONCLUSIONS AND FUTURE WORK

We used SPOT to identify hot spots in the criteria mentioned. Based on SPOT's findings (characterization and looking at videos), we proposed numerous design changes to RoboTutor. Table III presents some of the findings and their corresponding design implications.

In this paper we introduced SPOT, a tool to simplify data driven iterative design of an Intelligent Tutoring System by focusing on criteria such as reliability, recognition, usability and engagement. We described some hot spots that SPOT discovered and their design implications. SPOT can also be

Criterion	SPOT Findings	Design Implications
Reliability	Crashes especially occur in CountingX and Story Reading activities	Implement crash logging, Examine CountingX and Story Reading activities for bugs
Recognition	Children confuse between number pairs like 1 and 7 (often forget the middle dash in 7) and 5 and 3	Bias data sources of number copying and dictation to include such frequently confused digits for better practice
Usability	Children spend unusually long time per story in Story Reading activities	Add a timeout for story reading
Engagement	Children tend to bail out of an activity when given the same problems to solve repeatedly	Children should not be given the same problems repeatedly

TABLE III: SPOT Findings and Implications

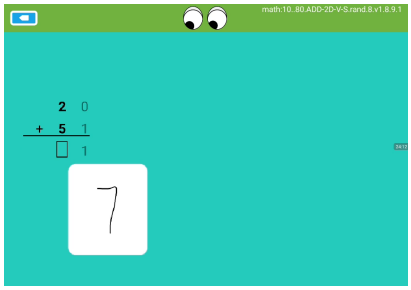


Fig. 5: Recognizer mis-recognizing 7 as 1 because of absence of middle stick - instance provided by SPOT

used for other Intelligent Tutoring Systems, since it only relies on very simple data sources.

As future work, we plan to develop SPOT into a web application and integrate it into LearnSphere [26], one of the worlds largest learning analytics platforms combining various tools for mining educational data from a variety of sources. We believe SPOT would benefit many tutor designers, help them iteratively improve their tutors, and in turn save a lot of time. We also plan to experiment with other scoring functions. Additionally, we wish to explore wheel-spinning [6] to discover if there are skills that children are not learning, as well as identify what design issues cause wheel spinning and can be eventually used as a SPOT criterion.

ACKNOWLEDGMENT

The work was partly done when S. Mian and M. Goswami were FICCI Research Scholars in the Robotics Institute Summer Scholars program at Carnegie Mellon University. They are thankful to Robotics Institute and FICCI for providing support and funding during the program to conduct this research.

REFERENCES

[1] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, pages 61–70. Springer, 2006.

[2] Chidanand Apté and Sholom Weiss. Data mining with decision trees and decision rules. *Future generation computer systems*, 13(2-3):197–210, 1997.

[3] Ryan Shaun Baker, Albert T Corbett, Kenneth R Koedinger, and Angela 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*, pages 383–390. ACM, 2004.

[4] Ryan SJd Baker. Modeling and understanding students' off-task behavior in intelligent tutoring systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1059–1068. ACM, 2007.

[5] Ryan SJD Baker, AMJA de Carvalho, Jay Raspat, Vincent Aleven, Albert T Corbett, and Kenneth R Koedinger. Educational software features that encourage and discourage gaming the system. In *Proceedings of the 14th international conference on artificial intelligence in education*, pages 475–482, 2009.

[6] Joseph E Beck and Yue Gong. Wheel-spinning: Students who fail to master a skill. In *International Conference on Artificial Intelligence in Education*, pages 431–440. Springer, 2013.

[7] Konstantina Chrysafiadi and Maria Virvou. Usability factors for an intelligent tutoring system on computer programming. In *New Directions in Intelligent Interactive Multimedia Systems and Services-2*, pages 339–347. Springer, 2009.

[8] Rehman Chughtai, Shasha Zhang, and Scotty D Craig. Usability evaluation of intelligent tutoring system: Its from a usability perspective. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 59, pages 367–371. SAGE Publications Sage CA: Los Angeles, CA, 2015.

[9] Mihaela Cocea, Arnon Hershkovitz, and Ryan SJD Baker. The impact of off-task and gaming behaviors on learning: immediate or aggregate? In *Proceeding of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*. IOS Press, 2009.

[10] Mihaela Cocea and Stephan Weibelzahl. Can log files analysis estimate learners' level of motivation? In *LWA*. University of Hildesheim, Institute of Computer Science, 2006.

[11] Mihaela Cocea and Stephan Weibelzahl. Cross-system validation of engagement prediction from log files. In *European Conference on Technology Enhanced Learning*, pages 14–25. Springer, 2007.

[12] Ryan SJ d Baker, Albert T Corbett, Kenneth R Koedinger, and Ido Roll. Generalizing detection of gaming the system across a tutoring curriculum. In *International Conference on Intelligent Tutoring Systems*, pages 402–411. Springer, 2006.

[13] Ryan SJ d Baker, Sujith M Gowda, Michael Wixon, Jessica Kalka, Angela Z Wagner, Aatish Salvi, Vincent Aleven, Gail W Kusbit, Jaclyn Ocumpaugh, and Lisa Rossi. Towards sensor-free affect detection in cognitive tutor algebra. *International Educational Data Mining Society*, 2012.

[14] XPRIZE Foundation. Global Learning XPRIZE. `http://learning.xprize.org`.

[15] Amrit L. Goel. Software reliability models: Assumptions, limitations, and applicability. *IEEE Transactions on software engineering*, (12):1411–1423, 1985.

[16] Andrina Granić and Vlado Glavinić. An approach to usability evaluation of an intelligent tutoring system. In *in N. Mastorakis, V. Kluev (eds), Advances in Multimedia, Video and Signal Processing Systems*, 2002.

[17] G. Henry, J. Lin, and C. Y. Park. SPOT: Refining robotutor. *HCI senior capstone project presentation, Carnegie Mellon University*, 2017.

[18] E Joseph. Engagement tracing: using response times to model student disengagement. *Artificial intelligence in education: Supporting learning through intelligent and socially informed technology*, 125:88, 2005.

[19] Nachimuthu Karunanithi, Darrell Whitley, and Yashwant K. Malaiya. Prediction of software reliability using connectionist models. *IEEE Transactions on Software Engineering*, 18(7):563–574, 1992.

[20] Kenneth R Koedinger, Julie L Booth, and David Klahr. Instructional complexity and the science needed to constrain it. *Science*, 342:935–937, 2014.

[21] Kenneth R Koedinger, Emma Brunskill, Ryan SJd Baker, Elizabeth A McLaughlin, and John Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

[22] Jack Mostow, Joseph Beck, Hao Cen, Andrew Cuneo, Evandro Gouvea, and Cecily Heiner. An educational data mining tool to browse tutor-student interactions: Time will tell. In *Proceedings of*

the Workshop on Educational Data Mining, National Conference on Artificial Intelligence, pages 15–22. AAAI Press, 2005.

[23] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[24] Google Play. AZ Screen Recorder. https://play.google.com/store/apps/details?id=com.hecorat.screenrecorder.free.

[25] Cristóbal Romero and Sebastián Ventura. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6):601–618, 2010.

[26] Carnegie Mellon University. LearnSphere. http://learnsphere.org/.

[27] Carnegie Mellon University. Project LISTEN. http://www.cs.cmu.edu/~./listen/.

[28] Carnegie Mellon University. RoboTutor. https://www.cmu.edu/scs/robotutor/.

Creating a Scalable Framework for Model-Free Reinforcement Learning in Norm-Rich Environments

Stephanie Milani
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250, USA
stephmilani@umbc.edu

***Abstract*— As autonomous robots make increasingly more decisions that significantly impact the lives of humans, it is essential for these agents to incorporate human norms in their decision-making processes. Robots can integrate these norms with traditional methods for learning how to accomplish tasks so that they can learn how to act in environments where actions can have social ramifications. To address the issue of scalability in previous work, we propose a novel framework, called penalty-modified Markov decision processes (PMMDP), for reinforcement learning in norm-rich environments. We formalize the learning and decision-making problem as solving a Markov decision process that is modified only as norms are violated. We show that our method has worst-case guarantees for the size of the state space that are equivalent to the average-case for previous methods.**

Keywords: model-free reinforcement learning, normative reasoning

I. INTRODUCTION

Emerging autonomous robots will make increasingly more decisions that significantly impact human lives. For that reason, these robots must incorporate human values and preferences into their decision-making processes. Robots can integrate these values and preferences - hereafter referred to as *norms* - with traditional methods for learning how to perform tasks so that they can learn how to act in environments where there may be social ramifications to actions. For example, the owner of a domestic robot may desire for it to not interrupt others while they are talking. If this robot that is tasked with communicating information to someone, but that has no sense of the "do not interrupt" norm, then it will not consider that norm when learning to accomplish its goal.

Previous attempts to integrate normative reasoning and task-completion methods suffer from a lack of scalability. In part, this issue stems from the coupling of norms and their resulting penalties, which means that norms that share the same penalties are considered to be separate, independent norms. In reality, penalties are often shared across various norm violations. For example, a robot may be sent to a particular room in the house for a "time-out" for a variety of different norm violations, but the end result is still the same: a "time-out". If the robot then violates another norm that leads to a "time-out" while on time-out in that room, there is no need to apply and keep track of that punishment twice.

In this work, we propose a novel framework, called *penalty-modified Markov decision processes* (PMMDPs), for reinforcement learning agents to learn how to act in an environment that incorporates norms. We formalize the learning and decision-making problem as solving a Markov decision process (MDP) that is modified only as norms are violated. We show that our method has worst-case guarantees for the size of the state space that are equivalent to the average-case for previous methods. We conclude by proposing future advancements to our method.

II. BACKGROUND

A Markov decision process (MDP) $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$ consists of the following components: a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a state transition probability function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ that defines the transition dynamics from one state to another, a reward function $R : \mathcal{S} \times \mathcal{A} \in \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$ that represents the difference in importance between present and future rewards. An MDP poses the problem to be solved by an agent, collectively referred to as a *domain*. The goal of a reinforcement learning (RL) agent is to improve its *policy*, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, which is defined as a probability distribution over state-action pairs, to achieve an optimal policy, π^*, that maximizes the cumulative expected rewards. Generally, an RL agent approximates one of two types of value functions in the process of learning a policy: the state-value function or the action-value function. The state-value function expresses the expected future discounted reward from a particular state, recursively computed over rewards received following the policy through state-action space as $V^\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s = s_t], \forall s \in \mathcal{S}$. The action-value function expresses the expected future discounted reward from a particular state given that that the agent takes a particular action, recursively computed over rewards received following the policy through state-action space as: $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s = s, a_t = a], \forall s \in \mathcal{S}$. An optimal value function V^* or Q^* can be derived from the optimal policy for the MDP.

A common means of solving for the optimal value function is to use *model-free reinforcement learning*, where the agent

tries to directly learn the optimal policy. Model-free reinforcement learning algorithms can be either *off-policy*, like Q-learning [1], or *on-policy*, like SARSA [2]. An off-policy learner updates its value function based on the maximum value of the next state, while an on-policy learner updates its value function based on the actions executed in the policy that is currently being followed. For this reason, on-policy algorithms tend to be more conservative with exploration, avoiding costly actions.

III. Related Work

In general, the majority of the papers in machine ethics consider agents whose primary and only task to reason ethically [3]–[5]. Prior work in norm-aware reinforcement learning uses reward shaping to incorporate human desires that are independent from the task-completion goals [6] or considers the issue of incorporating human values and preferences in a reinforcement learning agent by learning an ethical utility function that is a part of the hidden state of partially observable Markov decision processes [7]. However, in these approaches, only norms with reward penalties are considered, but long-term penalties, such as those that modify the components of the state space and/or the transition function, are not.

There is more work in constructing scalable computational frameworks for planning [8]–[10]. There exist two recent approaches that are most similar to ours. One approach, called *normative Markov decision processes* (NMDPs), explicitly represents norms and includes all possibly relevant norms in the state-space representation [11]. In a reinforcement learning setting, the use of this representation causes the problem succumb to the *curse of dimensionality* [12], an exponential explosion in the total number of states as a function of the number of state variables, which means that the problem is computationally intractable with a large number of norms. The *modular normative Markov decision process* (MNMDP) framework seeks to mitigate this issue by constructing a separate Markov decision process for each individual norm and for each set of interacting norms [13]. Notably, each individual MDP consists of the original components of the state space in addition to the relevant norm(s). This representation suffers from increased state-space size when there are a large number of interacting norms in the environment. In the case where there are a large number of interacting norms in the environment, using the NMDP approach is preferable to using the MNMDP approach because the total number of states is less in the former than in the latter, which we prove in Section VI.

IV. Penalty-Modified Markov Decision Processes

In this section, we detail our novel framework, first focusing on our representation for norms. We then discuss the different types of norms allowed by this framework. After that, we show how these norms are included in the MDP formalism, as well as how to modify the various components of the MDP to incorporate the penalties that arise from violating the norms.

A. Norm Definition

Here we depart from previous literature in our representation of norms by decoupling the changes that arise when an agent violates the norm from the norm itself, enabling the application of these modifications in the appropriate components of the MDP. In our framework, a norm n is represented as:

$$n \in \mathcal{N} = \langle C, \sigma \rangle \tag{1}$$

where C is the violation condition(s) and σ consists of the MDP modifications. The violation condition tests whether the norm has been violated; the modifications to the MDP represent the penalties that arise when a particular norm is violated.

1) Violation Condition: The violation condition C is a propositional function that determines whether a norm has been violated. It is defined as:

$$C : s_t \times a \times s_{t+1} \rightarrow [0, 1] \tag{2}$$

where s_t is the state from which the agent took the action a, and s_{t+1} is the resulting state. If a particular norm has been violated, it returns a 1; 0, otherwise.

The violations of the norm are conditioned on any number of state variables, and need not include all of the state variables. For example, the contents of the condition function for a norm dictating that a domestic robot tasked with delivering a message ought to not interrupt humans while they are speaking could test the following components of s_t:

- Is the person to whom the robot is delivering the message in a conversation?
- Is the message important?

in combination with testing if the robot took the action, speak, when in s_t. If all of the aforementioned conditions are true in s_t, then the agent is in violation of the norm and all of the modifications associated with violating that norm are applied to the MDP. Because these checks are conditioned on state variables and not whether a particular, pre-fixed state has been reached, norms can be applied to domains that share common attributes.

2) Violation Modifications: The set of violation modifications are the modifications to the MDP that are made as a result of violating a norm. There are three components in the set of modifications $\langle \sigma_R, \sigma_p, \sigma_T \rangle \in \sigma$, which represent modifications to the reward function, the state space (the penalty), and the transition function, respectively.

The first component modifies the reward function, imposing what is analogous to a fee if the agent has violated a norm. It represents the more immediate penalty for choosing not to follow the action prescribed by a given norm. It is represented by:

$$\sigma_R : r_t \rightarrow r_m \tag{3}$$

where r_t is the original reward received for taking the action and r_m is the new reward received based on the norm violation.

Algorithm 1 Modifying the MDP with Norms

function APPLY-PENALTIES(s_t, a, s_{t+1})
  for p in $\mathcal{N}$ **do**
    if PRE-EXISTING-PENALTY(s) $\vee$ IN-VIOLATION(s_t, a, s_{t+1}) **then**
      $s_{t+1} \leftarrow$ ADD-PENALTY-TO-STATE(s_{t+1}, $p.\sigma_p$)
function MODIFYING-THE-MDP-WITH-NORMS( )
  COMPUTE-TRANSITION(s_t, a, s_{t+1}) ▷ compute for unmodified state
  APPLY-PENALTIES(s_t, a, s_{t+1})
  COMPUTE-NORM-MODIFIED-TRANSITION(s_t, a, s_{t+1}) ▷ compute for norm-modified state
  return s_{t+1}

The second component consists of a penalty flag that modifies the state space to indicate that a particular violation has occurred. It is represented by:

$$\sigma_p \rightarrow 1 \tag{4}$$

and is a variable that takes on a single value, 1, corresponding to the violation of a particular norm. If the agent takes an action to violate the norm, then the associated penalty flag is added to the resulting s_{t+1}, as well as all future states encountered in the episode.

The inspiration for this type of penalty is from how penalties are applied in the real world: if someone takes an action that results in a violation of a norm or a law, then the penalties are represented as separate entities. For example, a robot can either enter the bedroom or not, which is separate from other potential consequences, like a ban from entering a bedroom, a ban from touching a particular item, and so on.

The third component modifies the transition function, taking the agent to a new state after a norm violation. It is represented by:

$$\sigma_T : s_{t+1} \rightarrow s_{t+1} \cup \sigma_p \tag{5}$$

where s_{t+1} is the original resulting state from the transition function and $s_{t+1} \cup \sigma_p$ is the new resulting state that includes the penalty for the norm violation.

It represents the long-term penalty that arises from violating a norm. For example, suppose the penalty for the robot being rude to guests is a "time-out" to a particular room in the house. If an agent violates the politeness norm and is restricted to the "time-out" room, then the result is an addition of penalty flag that modifies the state space, as well as a change in, at a minimum, the x and y coordinates of the agent to reflect the movement of the robot to the "time-out" room. Until the end of the episode, the agent is then unable to move from the "time-out" location.

These transition modifications can alternatively be used to help guide the agent along the right path after a mistake. For example, suppose there is an agent that is attempting to navigate to a particular area. While attempting to reach that area, the agent violates a norm by choosing some exploratory action. Instead of imposing a transition modification as a punishment, the transition modification is applied to constrain the agent's x-coordinate to one that is shared with the goal location, enabling it to reach the goal faster while making fewer mistakes.

B. Types of Norms

There exist two types of norms: those that are implicit and those that are toggleable. Implicit norms are norms that are baked into the MDP and cannot be turned off. We primarily focus on this type of norm in this work.

Toggleable norms are dependent on some feature of the MDP being turned on. This representation is akin to a condition function where if some violation has already occurred, then the penalty is compounded. Suppose a robot is already receiving some punishment, represented as a penalty in the state space, for taking an undesirable action. It then takes another action that violates a norm, where the violation of that norm depends on the robot receiving the aforementioned punishment. Notably, if the agent were to have taken that same action from that same state, sans the norm penalty, then it would not be in violation of that norm. This capability enables norms to be conditioned on features of the environment that may change, such as visitors who may be going in and out of the home.

C. The Penalty-Modified Markov Decision Process

A penalty-modified Markov decision process (PMMDP) is a six-tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma, \mathcal{N} \rangle$ consisting of the following components: a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, a reward function $R : \mathcal{S} \times \mathcal{A} \in \mathbb{R}$, an ordered set of norms $\mathcal{N}$ applicable in the state space, and a discount factor $\gamma \in [0, 1]$. The ordering of the norms is crucial for producing a principled way to apply the penalties. The ordering represents the importance, or priority, of the norms and the resulting penalties.

V. Learning in an Environment with Norms

For an agent to learn in an environment with norms, the penalty-modified MDP can either be precomputed or modified during the learning process. The application of norm penalties to the MDP proceeds is shown in Algorithm 1. When an agent takes an action in an environment, the original transition is computed, sans norms. Using the set of norms, the environment checks for any resulting penalties from norm violations or whether the agent violated a norm by taking action a from state s_t. If there are existing or new penalties, then the penalties are applied to the corresponding aspect of the MDP: the state, the transition function, and/or the reward function. Then, the transition is recomputed including modifications to the state space.

VI. State-Space Size Analysis

In this section, we show that, by using our definition of norms and the resulting MDP, we do not encounter the same state-space explosion as previous methods. We show the benefit of our method by, first, showing that using there exist some cases where using the NMDP approach is preferable to using the MNMDP approach. Next, we show that our method is always at least as good as the MNMDP and NMDP approach in terms of the size of the state space. With our approach, the complexity of the MDP is only increased with the number of penalties that are imposed by the norm violations. If the agent does not violate any norms, then no penalties are imposed, and, thus, the agent solves the original MDP, making the best-case size of the state space equivalent to that of the original, normless MDP. Importantly, this analysis is only for binary norm features; future work will extend this method to handle non-binary features. See Table I for an overview of our analysis.

Theorem 1. *There exist some cases where using the NDMP approach is preferable to using the MNMDP approach.*

Proof. Let n be the total number of norms. Let d be the total number of possible interactions between the norms. Then, as claimed [13], the number of states in the fully normative MDP framework is of the order $\Omega(2^n)$ and the number of states in the MNMDP framework is of the order $\Omega(n^d)$. Thus, using the MNMDP framework is preferable if $n^d < 2^n$, which can be converted to $log_2 n^d < n$ by taking the log of both sides. Then, $d(log_2 n) < n$, so $d < n/log_2 n$. Thus, the MNMDP framework is preferable to the fully normative MDP approach when the number of interactions is less than $n/log_2 n$.

Theorem 2. *Our method is always as good as the MNMDP and the NMDP approaches.*

Proof. Let p be the number of possible penalties. In the worst case of a one-to-one mapping of penalties to norms, where each norm has its own unique penalty, the number of penalties p is the same as the number of norms n ($p = n$); however, in reality $p \leqslant n$ because the set of norms to penalties consists of one-to-one or many-to-one mappings. Let $|P|$ be the number of possible concurrent penalties. Because $p \leqslant n$, $|P| \leqslant 2^{|P|} \leqslant 2^n$. Continuing with the aforementioned notation, let d be the maximum number of concurrent norms. Then,

$$|P| = \sum_{i=1}^{d} \binom{p}{d} = \binom{p}{1} + ... + \binom{p}{d} \rightarrow p^1 + ... + p^d \quad (6)$$

Hence, the number of states created in our approach is of the order $O(p^d)$, meaning that our worst-case bound is equivalent to the average-case of previous methods.

Approach	Number of States (Order of)	Note
NMDP	2^n	
MNMDP	n^d	If $d > n/lg(n)$, more states created than NMDP.
PMMDP	$2^p, p^d$	$p \leqslant n$

TABLE I: Here is the state-space size analysis for each of the three compared methods. The state-space size for NMDPs is exponential in *n*, where *n* is the number of norms that can be either on or off. The state-space size for MNMDPs is polynomial in *d*, where *d* is the maximum number of interactions between norms. The worst-case state-space size for PMMDPs is exponential in *p* or polynomial in *d*; however, importantly, $p \leqslant n$, so no more states are created than in the other two methods.

VII. Conclusion

We present a novel framework for incorporating normative and task-based reasoning for reinforcement learning that can modify states, transitions, and rewards. The number of states in our framework is less than other frameworks because it depends on the number of penalty flags, not on the number of norms ($p \leqslant n$). The computed worst-case bound on the number of states in our method is of the same order as the average cases for previous work. Our framework also avoids redundant states, adding no more states than are included in the naive case.

VIII. Future Work

Our complexity bounds are not tight; the actual performance of our framework will likely be better in practice. We will empirically evaluate our method to better estimate actual performance. We will conduct this evaluation on the scenarios in previous work to empirically validate our claims and provide a better comparison to existing frameworks.

Because domestic service robots are a promising application of normative reasoning due to the rich and varied norm preferences and representations, we will then use our novel environment that we created using BURLAP, *Household*, to represent our target test environment of domestic service robots. In Household, which is loosely based off of Cleanup [14] an agent navigates through rooms and doors to reach a desired goal. The agent can move north, south, east, or west; it can also attempt to open doors, which can either be locked or unlocked. If a door is locked, then the agent

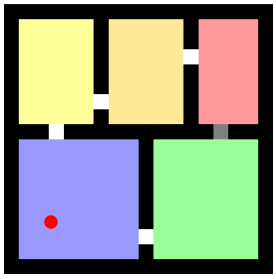


Fig. 1: An example 18x18 household environment with five doors and five rooms. Gray doors are locked; white doors are unlocked. The red dot in the blue room is the agent. The goal of the agent is to reach its target destination while violating the minimum amount of norms.

cannot pass through it; if a door is unlocked and open, then the agent can choose to pass through it. Figure 1, shows an example configuration of an 18x18 grid consisting of five rooms and five doors. We will use a number of different configurations of the Household environment, as well as a variety of norms, to test the efficacy of our method in our target application area.

ACKNOWLEDGMENT

Stephanie would like to thank the National Science Foundation Robotics Institute REU for funding this work. She would especially like to thank Dr. Katia Sycara for her guidance and feedback. She would also like to thank Vignesh Krishnamoorthy and Nicholay Topin for their advice, as well as Rachel Burcin, John Dolan, and the rest of the Robotics Institute Summer Scholars team.

REFERENCES

[1] C.J. Watkins & P. Dayan, Q-learning. *Machine Learning* 8(3-4), pp. 279-292, 1992.

[2] R. S. Sutton & A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[3] M. Anderson & S. L. Anderson, Machine ethics: Creating an ethical intelligent agent. *AI Magazine* 28(4), p. 15, 2007.

[4] R. C. Arkin, Ethics and autonomous systems: Perils and promises [points of view]. In *Proceedings of the IEEE* 104(10), pp. 1779-1781, 2016.

[5] S. Bringsjord, K. Arkoudas, & P. Bello, Toward a general logicist methodology for engineering ethically correct robots. *IEEE Intelligent Systems* 21(4), pp.v38-44, 2006.

[6] Y.-H. Wu & S.-D. Lin, A low-cost ethics shaping approach for designing reinforcement learning agents. In *AAAI*, pp. 1687-1694, 2018.

[7] D. Abel, J. MacGlashan, & M. L. Littman, Reinforcement learning as a framework for ethical decision making. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[8] F. Meneguzzi, O. Rodrigues, N. Oren, W. W. Vasconcelos, & M. Luck, BDI reasoning with normative considerations. *Engineering Applications of Artificial Intelligence* 43, pp. 127-126, 2015.

[9] N. Oren, W. Vasconcelos, F. Meneguzzi, and M. Luck, Acting on norm constrained plans. In *International Workshop on Computational Logic in Multi-Agent Systems*, Springer, pp. 347-363, 2011.

[10] S. Panagiotidi & J. Vázquez-Salceda, Norm-aware planning: Semantics and implementation. In *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pp. 33-36, 2011.

[11] M. S. Fagundes, S. Ossowski, J. Cerquides, & P. Noriega, Design and evaluation of norm-aware agents based on normative Markov decision processes. *International Journal on Approximate Reasoning* 78, pp. 33-61, 2016.

[12] R. E. Bellman, & S. E. Dreyfus, Applied dynamic programming, 4th edn., Princeton University Press, 1971.

[13] V. Krishnamoorthy, W. Luo, M. Lewis, & K. Sycara, A Computational Framework for Integrating Task Planning and Norm Aware Reasoning for Social Robots. In *IEEE International Conference on Robot and Human Interactive Communication*, 2018.

[14] J. MacGlashan, M. Babeş-Vroman, M. desJardins, M. L. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, & L. Yang, Grounding English commands to reward functions. In *Robotics: Science and Systems*, 2015.

Realistic Simulation System for Environmental Understanding of Autonomous Driving Vehicles

Akari Minami[1], Zhiqian Qiao[2], John M. Dolan[3]

***Abstract*— Currently, the behavior of autonomous vehicles tends to be unpredictable, which can affect the behavior of other vehicles and people around them. To improve this and make autonomous vehicles drive similarly to human driving, many researchers are working on algorithms. In this project, we use Inverse Reinforcement Learning (IRL). Applying IRL, we need to collect human driving data to get reward functions. For this, we chose using a simulator over driving with actual cars for convenience. We asked subjects to drive in a simulator with a steering wheel and pedals to obtain the data, making this human experience as realistic as possible. In this paper, we document the driving interface with the simulator and the data collection process.**

I. Introduction

A. Current Problem of Autonomous Driving

Autonomous vehicles work very well in tasks such as obstacle avoidance, lane-keeping and car-following. However, when merging into another lane, such as occurs in intersections and lane changing, the way they drive is not natural compared to actual human driving. As they predict other cars' future trajectories and try to stay out of their way, they tend to be too defensive and indecisive. For instance, they wait until surrounding cars go away, block two lanes when they want to switch to a third lane, and suddenly stop in the middle of an intersection when a pedestrian enters a crosswalk even though there is enough distance to finish turning. This can affect other vehicles' behavior in the real world. For example, they may simply block a human driver's way and cause them to stop/slow down.

B. Need for Simulation

Many researchers are working on algorithms to improve these behaviors. Human driving data are a good source for deriving a competent autonomous driving model, but it is difficult to formulate the reward function for driving (for Reinforcement Learning) and we want autonomous driving to be applicable in environments it has not experienced (for imitation learning). Hence, we chose to use Inverse Reinforcement Learning (IRL) in this research. To apply IRL, we need to collect human driving data to generalize a human driver's behavior model and to find a preferred reward function. There are two ways to collect data: actually driving in a town with a real car and driving in a simulator. In the actual driving approach, you need to put cameras and sensors on the car, and you might need to care about specific traffic conditions or go to places with an ideal environment [2]. On the other hand, a simulator doesn't require you to go somewhere else and enables you to create an environment. As we need to obtain a sufficiently large amount of human driving data to train autonomous vehicles, which is time-consuming and laborious, we decided to use a simulation for its convenience. There is also a safety concern with testing algorithms in the real world since they might behave in unexpected ways. Thus, we would like to use a realistic simulation for this human data collection stage and to simulate driving policies derived from the reconstructed cost function as well [1]. In this project, we set an urban area as a target environment.

C. Our Goal

Our final goal is to make autonomous vehicles drive as humans do using IRL. We will apply the obtained reward function of the human behavior model to the behavior of an autonomous vehicle itself. As a first step, we set the goal for this project to make the human experience in simulation as realistic as possible and to collect human driving data. For data collection, we asked human subjects to drive a car in the simulation using a steering wheel and pedals (Figure 1). Making the response of the simulator to human action comparable to actual driving is essential in order to make a subject feel like he/she is driving a real car with its wheel and pedals.

Fig. 1. Driving in a simulator with a steering wheel and pedals

[1]Akari Minami is a 2018 Robotics Institute Summer Scholar and is a Mechanical Engineering undergraduate at Kyushu University, Japan. This work was supported by TOBITATE! Young Ambassador Program.

[2]Zhiqian Qiao is a PhD student at Robotics Institute in Carnegie Mellon University.

[3]Dr. John M. Dolan is with the Robotics Institute at Carnegie Mellon University

D. Related Work

Broadly speaking, an objective of this research is to make the behavior of autonomous driving better using IRL. This section will introduce previous research work using IRL.

1) Using human driving model as a model of vehicles around an autonomous vehicle: In both our project and this work, we use IRL to learn a reward function which enables us to mimic human driving. The differences are in the objective and the object modeled by the reward function. Our objective is to make an autonomous vehicle behave like a human-driven vehicle, so we derive the reward function of the autonomous vehicle itself. On the other hand, in [1], they wanted autonomous driving to be more effective by enabling an autonomous vehicle to leverage effects on human actions, so they applied the reward function to human-driven vehicle models around the autonomous vehicle. In the model they assume, the interaction between an autonomous vehicle and a human driver is dynamic system, in which the autonomous vehicle's action has immediate effects on human action as well as the vehicle's state. They proposed a dynamics model for this system by modeling the human as optimizing the reward function they got from IRL. In case studies, they gave an autonomous vehicle reward function with a specific desired human action, and the behavior of the trained autonomous vehicle actually affected the human model's behavior in the way they expected. After that, they did user studies, in which a subject drove a targeted vehicle with a trained autonomous vehicle, and the behavior of the autonomous vehicle could purposefully modify human behavior. These results show autonomous vehicles can generate human-interpretable behaviors through optimization, which uses a reward function learned through IRL [1].

II. Data Collection

A. Simulator

We chose the CARLA simulator and Logitech G920 (steering wheel and pedals) for this project. CARLA is an new open-source simulator for autonomous driving research released on March 27, 2018. In addition to open-source code and protocols, CARLA provides open digital assets (buildings, vehicles, pedestrians, traffic lights, streets signs) we can use freely, which allows the autonomous vehicles to run in a realistic environment. The environment has 2 default town maps (you can create a new one as well) and 15 weather presets, and you can also set the number of other vehicles and pedestrians [3].

B. Controlling Ego Vehicle with Steering Wheel and Pedals

In order to collect human driving data by letting subjects drive in simulation, we interfaced the simulator with the steering wheel and pedals. To make driving in simulation as close as possible to real driving, we set the right pedal as the accelerator, and the middle pedal as the brake. The remaining third pedal is not used in this project since we assume an automatic transmission. The Logitech pedal outputs are 1 when not pressed and -1 when pressed all the way down, but CARLA expects a [0, 1] range with zero acceleration for the input 0, so the sign and the range of the pedal input were changed as shown in TABLE 1.

TABLE I
Description of the Logitech input

	Function	Description
right pedal	throttle	Throttle input between [0.0, 1.0]
middle pedal	brake	Brake input between [0.0, 1.0]
wheel	steer	Steering input between [-1.0, 1.0]
LB plate	toggle reverse	T/F (In reverse gear when True)
B button	hand-brake	T/F (Hand-brake is engaged when True)
X button	toggle autopilot	T/F (In autopilot mode when True)
Y button	change episode	Save graph and start new episode

For conversion of the throttle input from the pedal (I_{pt}) to the simulator (I_{st}):

$$I_{st} = (-I_{pt} + 1.0) \div 2.0$$

The Logitech brake is stiffer than a brake in an actual car, so the brake input from the pedal (I_{pb}) was converted to the brake input to the simulator (I_{sb}) as follows:

$$I_{sb} = (1.0 - I_{pb}) \times 3$$

Since the steering was not responsive enough with the original steering input (I_{ws}), the steering ratio was changed to depend on the steering level (I_{ss}):

$$I_{ss} = \begin{cases} I_{ws} \times 2.0 - 0.08 & (I_{ws} < -0.1) \\ I_{ws} \times 3.0 + 0.02 & (-0.1 \leq I_{ws} \leq -0.01) \\ I_{ws} & (-0.01 < I_{ws} < 0.01) \\ I_{ws} \times 3.0 - 0.08 & (0.01 \leq I_{ws} \leq 0.1) \\ I_{ws} \times 2.0 + 0.08 & (0.1 < I_{ws}) \end{cases}$$

C. Data Item

As subjects drove, we collected the following data at each step:

- RGB image from default camera: view of scene
- Raw image from depth map camera: depth of objects in a view (Max render distance: 1 km ahead)
- Raw image from semantic segmentation camera: view classified with tag information
- Sensor data from ray-cast-based rotating LIDAR: distance between the car and surrounding objects
- Driving data of the human-driven vehicle: time step, time, input from a wheel and pedals (steer, throttle, brake), speed, acceleration, transformed location (x, y, z), transformed orientation (x, y, z), transformed rotation (pitch, yaw, roll)
- Driving data of other vehicles in the town: time step, time, vehicle id, speed, transformed location (x, y, z), transformed orientation (x, y, z), transformed rotation (pitch, yaw, roll)

All of the images are obtained from the driver's perspective, which is the same as what subjects see while driving. A graph of shifts in input from wheel and pedals, speed, and acceleration over time can be shown at the same time as the subject is driving and saved as an image at the end of each episode. This graph shows speed and acceleration of the human-driven vehicle corresponding to throttle/brake pedal

input against time. The images and the graph are shown in Figure 3.

III. User Studies

We conducted user studies and collected human driving data. The focus is on the trajectory and the way the driver accelerates/decelerates at curves as well as according to traffic lights and speed limits. As we want to see basic driving behavior of human, there is no interaction with other vehicles and pedestrians in this study.

A. Settings and Configuration of CARLA

We used the following settings for CARLA: Town01 map, clear noon for weather, no other vehicle and pedestrian. The system gives a view from the driver's perspective. In order to keep up with real-time, we used a variable time-step and ran the simulation in synchronous mode.

B. Condition

We asked 4 participants (1 female, 3 male), who had driver's licenses and more than 2 years of driving experience, to drive in the simulator with a steering wheel and pedals (Figure 1), and collected 8 sets of driving data. The drivers followed traffic lights and speed limits as much as possible and drove for 2.5 minutes, which is long enough to encounter and go through both curves and intersections.

IV. Discussion

In this section, we compare human driving with the driving of the autopilot, which is the hard-coded AI inside CARLA. Figure 4 shows driving data of one of the human subjects and the autopilot for about the first 1100 steps along different routes.

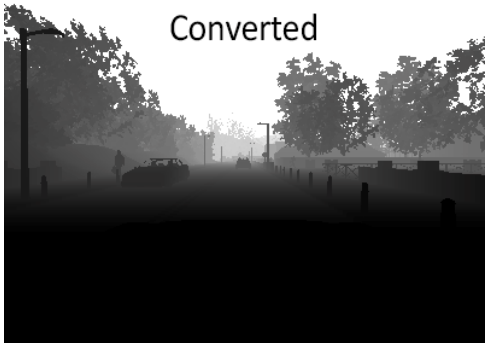
- The route of the subject: 60 km/h, intersection (turning right), 30 km/h, 60 km/h, curve, 90 km/h, curve, 60 km/h, traffic light (red)*
- The route of autopilot: 30 km/h, curve, 30 km/h, traffic light (red), 30 km/h, intersection (turning left), 30 km/h, 90km/h, 30km/h, traffic light (red)*

*30 km/h, 60 km/h, 90 km/h are straight roads with speed limits of 30 km/h, 60 km/h, 90 km/h.

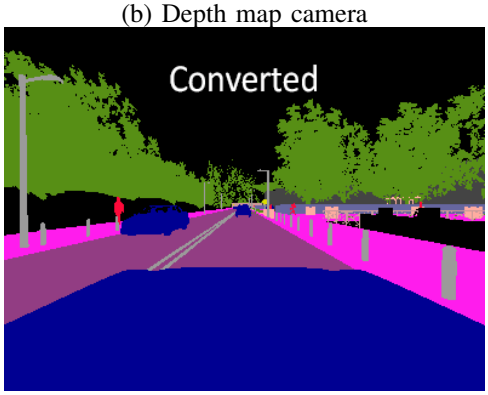
There are two big differences between those two types of driving. The autopilot drastically accelerates/decelerates. Looking at Figure 4 (b), the slope of the speed is nearly vertical when it tried to change speed based on speed limit, no matter how fast the limit is. On the other hand, human drivers change speed more smoothly. Furthermore, the autopilot does not slow down at all when it turns at intersections and curves. As it's designed to drive at the right speed depending on speed limit, it keeps going at the speed. However, human drivers slow down because it's physically hard to make a turn at high speed in the real world. That's why the speed of the human-driven vehicle drops to 10-20 km/h even though the speed limit after a curve is larger than one before the curve.

Although this autopilot is not as sophisticated as an advanced autonomous driving model, this comparison of

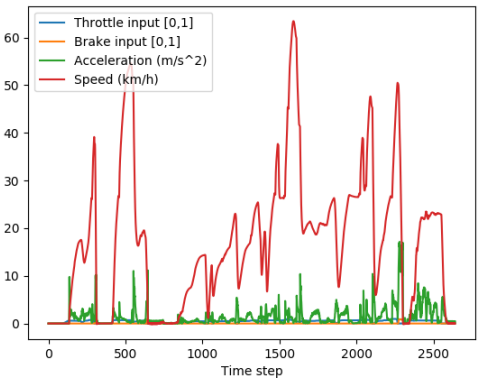
(a) Default camera



(b) Depth map camera



(c) Semantic segmentation camera



(d) Human driving data

Fig. 2. (a) - (c) are images obtained from cameras, (b) and (c) are images converted to a more human-readable palette of colors [3], (d) shows speed and acceleration of the human-driven vehicle corresponding to throttle/brake pedal input against time.

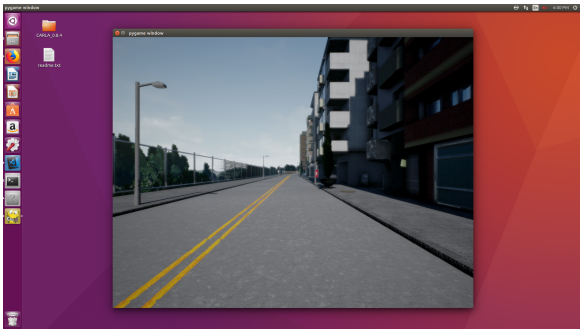


Fig. 3. Screen which subjects see while driving

driving data shows big gaps between the designed driving policy and human driving policy. We need to take physical and environmental conditions into account as well as driving techniques when we design the driving policy of autonomous vehicle, and learning from actual human driving data can provide significant insights.

V. Future Work

In this project we developed a realistic simulation environment and collected human driving data. As mentioned in the introduction, our goal is to train autonomous vehicles to drive as humans do. Once we collect more human driving data, both information data and image data could be used to find a preferred reward function for the following actions:

- trajectory and the way the driver accelerates/decelerates at curves (with/without oncoming car)
- the driver's acceleration/deceleration according to the color of traffic lights and speed limits
- how the driver reacts to other vehicles at intersections
- how drivers react to pedestrians and cyclists
- how a driver's view affects the driver's behavior (if the driver can recognize others before reaching an intersection)
- relationship between what the driver sees and the driver's action

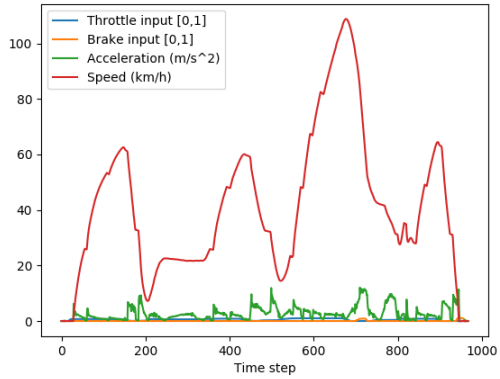
We can work on vehicle detection and tracking with images and sensor data, and on learning of human driver behavior with human driving data.

VI. Acknowledgment

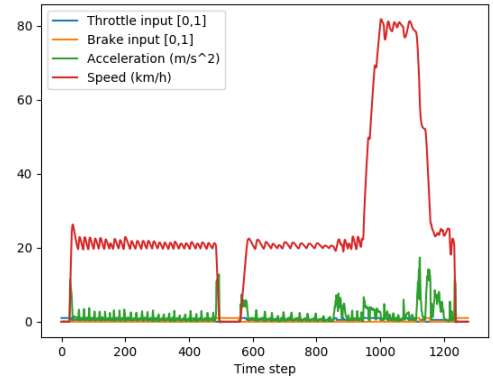
I am grateful to Dr. John M. Dolan for his mentorship and guidance throughout the summer, and to Zhiqian (Carolyn) Qiao and other lab members for their help. Huge thanks to Rachel Burcin for everything she has done for us and making this amazing RISS program happen, and to the whole RISS cohort for huge support. I would like to thank TOBITATE! Young Ambassador Program of Japanese government for funding my research.

References

[1] Dorsa Sadigh, Shanker Sastry, Sanjit A. Seshia, Anca D. Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. In Proceedings of the Robotics: Science and Systems Conference (RSS), June 2016.



(a) Human driving



(b) Autopilot driving

Fig. 4. Speed, acceleration, and throttle/brake pedal inputs of the human-driven vehicle and autopilot-driven vehicle against time for first 1100 steps.

[2] Masamichi Shimosaka, Kentaro Nishi, Junichi Sato, Hirokatsu Kataoka. Predicting driving behavior using inverse reinforcement learning with multiple reward functions towards environmental diversity. In Proceedings of 2015 IEEE Intelligent Vehicles Symposium (IV 2015), pp. 567572, 2015.

[3] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lpez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Conference on Robot Learning (CoRL), 2017.

Real-Time Semantic Segmentation of Sparse LiDAR Point Clouds using SqueezeSeg and Recurrent CRF*

Ingrid Navarro
ingrid.navarro.an@gmail.com
Department of Engineering and Information Technology
Tecnológico de Monterrey

Luis E. Navarro-Serment
lenscmu@ri.cmu.edu
The Robotics Insitute
Carnegie Mellon University

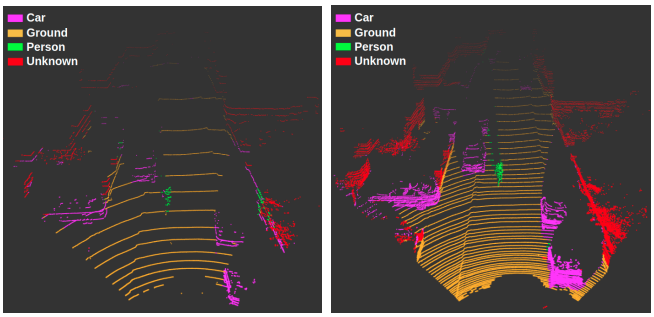
***Abstract*— The use of Deep Learning for semantic segmentation of highly sparse point clouds has not been fully explored. Two key reasons are the unavailability of training data, and the fact that high levels of sparsity make it difficult to interpret object structure. In this paper, we explore semantic segmentation in urban environments of 3D point clouds that present high-levels of vertical sparsity. We leverage our work on a lightweight CNN called *SqueezeSeg*, which was trained on the KITTI dataset for the object instances *car*, *pedestrian* and *cyclist*. We extend SqueezeSeg to include the *ground* class, which is not currently available in this dataset. Similarly, the point clouds from this dataset exhibit high vertical density, however, our focus is on processing point clouds of lower vertical density generated by low-end scanners. In this work we first developed an approach to obtain annotated ground examples from the KITTI dataset. Then the examples from the dataset were downsampled to produce sparse point clouds while retaining the annotations of the original dataset. Subsequently, we trained the CNN with the sparse data using Recurrent Conditional Random Fields (CRF) to increase the accuracy of predictions. Our experiments show that the extended CNN achieves prediction accuracies which are comparable to the original approach, which was trained on denser point clouds.**

***Index Terms*— Semantic Segmentation, Sparse Point Clouds, Deep Learning**

I. INTRODUCTION

Autonomous vehicle navigation depends on real-time perception systems that produce accurate descriptions of the scenes and the objects within it. With the increasing computing capabilities enabling continuous progress in Deep Learning algorithms, these systems are becoming highly robust and capable of making descriptions of the environment with high levels of granularity. In particular, LiDAR sensors have become one of the most prevalent components of these perception systems. Such sensors are able to collect highly accurate positional measurements that are stored in a data structure called *point cloud*. These measurements allow to directly model the geometry of the environment more accurately than vision-based sensors. Nevertheless, high levels of analysis and costly computation resources are often required to extract meaningful information from the point clouds because the span between adjacent laser pulses generated by these sensors grows as they travel farther from it, producing sparse data that is difficult to interpret.

*This work was supported by Tecnologico de Monterrey and Carnegie Mellon University



(a) Sparse Point Cloud **(b)** Dense Point Cloud

Fig. 1: Example of segmentation results of a sparse point cloud (a) compared against a dense point cloud (b).

Previous work on scene interpretation from LiDAR point clouds includes multi-stage processes that require numerous steps such as ground removal, clustering-based segmentation and manual feature mining [1], [2], [3], [4]. These approaches rely on hand-crafted and complicated features [5] or even random initializations [6], all of which are usually unable to gather a general context of the environment and might exhibit significant instability.

Motivated by their success in other fields like Computer Vision, there has been a rising interest in designing Deep Learning algorithms to perform tasks such as instance segmentation [7], [8], [9], object detection [10] and reconstruction [11] of 3D Point Clouds. However, these approaches often require to incorporate image-based sensors to produce a better interpretation of the point cloud data. Moreover, such approaches are usually tailored to 3D scanners that produce highly dense representations of the environment (e.g Kinect, MatterPort or Velodyne HDL-64E, illustrated in Fig. 1-(b)). However, our particular motivation is to explore semantic segmentation in point clouds presenting higher levels of vertical sparsity, such as those produced by a Velodyne VLP-16 sensor, as seen in Fig. 1-(a), since this type of sensor is less expensive in terms of cost and computation budget compared to other LiDAR sensors like the Velodyne HDL-64E.

To our knowledge, approaches that directly operate on this kind of point clouds using Convolutional Neural Networks (CNN), have been seldom studied. Thus, working with

highly sparse point clouds[1] remains a largely open topic in perception systems.

For this reason, we explore the potential of Deep Learning to perform semantic segmentation of these point clouds. In particular, we focus on segmenting dynamic outdoor scenes from point clouds in urban environments. First, we adapt the SqueezeSeg [9] CNN which is currently able to segment cars, pedestrians and cyclists from the KITTI dataset [12] to operate on sparse point clouds. Then, as we are limited by the unavailability of training data, we leverage on the LiDAR scans from this dataset which already contains labels of the aforementioned classes. However, our focus includes performing ground segmentation, but annotating data manually is rather labor intensive and time-consuming. Thus, to generate training samples for the *ground* class, we designed an automatic ground annotator tool using a 3D plane-fitting algorithm to estimate the ground. Next, we designed a method to down-sample the KITTI dataset by reducing the vertical density of the point clouds to simulate the scans that our sensor of interest (i.e. Velodyne VLP-16) would produce. Finally, we used the generated dataset to train the CNN and evaluated the performance of the model using Recurrent Conditional Random Fields to refine the inference maps of the sparse point clouds. We compare our experiments against the original approach and show that our experiments attain accurate results.

II. RELATED WORK

A. Segmentation of 3D Point Clouds

Previous work show a wide range of approaches for ground extraction and cluster-based segmentation. In [2], they perform real-time ground filtration of sparse LiDAR scans by projecting the angles between vectors onto a 2D plane and interpolating data using iterative algorithms for ground estimation such as Random Sample Consensus Algorithm (RANSAC) [6], [13]. The authors in [4] propose a two-step segmentation pipeline for LiDAR scans from the KITTI dataset. First, they use a plane-fitting technique to extract the points belonging to the ground. Then, they cluster the remaining points using a connected component approach. In[1], they propose fast region-growing algorithm that maps sparse point clouds into range images and establishes a relationship between points and the distribution of the reflectivity in a particular object instance. Bogoslavski and Stachniss [14] explore different algorithms for ground segmentation, clustering based on local neighborhoods and clustering based on vertical variance. The former approach might not perform well when evaluated in cluttered environments, whereas the latter might fail when dealing with sparser and unstructured data.

In general, although these approaches are usually able to perform fast ground segmentation and clustering, they present underlying disadvantages. First, they are usually tested on environments where conditions are ideal and well-structured. Second, approaches that depend on random initializations or iterative algorithms to perform ground removal or object clustering might be subject to instability. Thus, when evaluated in cluttered environments, these approaches might require extensive pre-calibration, optimizations and multi-stage processes that depend on highly complex feature mining algorithms [3], [15] to increase the robustness against instability.

B. Semantic Segmentation of Point Clouds using CNNs

The foregoing limitations have motivated the study of robust perception systems for point clouds using Convolutional Neural Networks. Velas et al. [8] present a fast method to perform binary classification of Velodyne VLP-64 point clouds from the KITTI dataset, into ground and non-ground points using different shallow CNNs. They encode the point cloud into a dense 2D matrix, and then use a height difference constraint to separate ground points from non-ground points. Then they train the data using four different model architectures.

One of the current state-of-the-art approaches for point cloud segmentation is described by Qi et al. [16]. They designed a novel neural network called PointNet that is invariant to point cloud permutation. Furthermore, their approach performs both, object classification and semantic segmentation. Leveraging their work on the PointNet architecture, Engelmann et al. [17] explored mechanisms to incorporate spatial context into the network architecture like neighborhood information and point descriptors. In [7] the authors explored using 3D Fully Connected Conditional Random Fields with neural networks to obtain fine-grained semantic segmentation of point clouds.

Wu et al. [9] addressed the point-wise semantic segmentation of instances such as cars, pedestrians and cyclists from raw point clouds obtained from the KITTI dataset. They designed and end-to-end pipeline called *SqueezeSeg* that first projects the point cloud onto a sphere to produce a compact representation of the data. Then, it uses a CNN to produce an output point-wise labeled map, and finally it refines the predictions using Recurrent Conditional Random Fields (CRF) as in [18]. The latter, has been successfully used in several image-based semantic segmentation approaches [19], [20], [18] to restore details that might be lost in down-sampling operations.

III. APPROACH

In the following subsections we introduce the methodology followed in our work. In subsection A, we review the CNN used in our approach [9] and how it was adapted to operate with our data of interest.

As described before, there are two main aspects addressed in our approach. First, extending the capabilities of the network to perform ground segmentation together with cars, pedestrians and cyclists[2]. Second, obtaining the appropriate

[1]In this work, when mentioning sparse point clouds we refer to sparsity in the vertical direction

[2]In our experiments, we merged the instances for pedestrians and cyclists into a single class

training data, and evaluating the performance of the network when it operates with it. To address the former, we implement an algorithm to label the ground points in the point clouds from the original dataset. For the latter, we propose the down-sampling method to reduce the vertical density of the scans in the original dataset and generate a dataset of sparse point clouds. We present the details of our implementation of both methods in subsections B and C, respectively.

A. SqueezeSeg and Conditional Random Fields

Our work is based on the Convolutional Neural Network structure SqueezeSeg [9], which in turn is derived from the lightweight CNN SqueezeNet [21]. The network is designed to work like conventional CNN models that operate on images of size H x W x C, where H and W encode the spatial position, i. e. the height and width, and C encodes the features per point. To use this convention, they transform the point cloud into a more compact representation by projecting it onto a 2D spherical grid (θ, ϕ) obtained as

$$\theta = \arcsin \frac{z}{\sqrt{x^2 + y^2 + z^2}}, \tag{1}$$

$$\phi = \arcsin \frac{y}{\sqrt{x^2 + y^2}} \tag{2}$$

where θ represents the azimuth and ϕ is the zenith angle. Then, the angles are discretized using the resolutions of vertical spanning ($\Delta\theta$) and rotation ($\Delta\phi$). The original work carried out their experiments using the LiDAR data from KITTI, which was collected from a Velodyne HDL-64E sensor with 64 vertical channels. The number of channels represents the height of the 2D spherical projections, thus, H = 64. Then, the 2D spherical projection represents points within a 90° front-view of the point cloud discretized into a 512 grid. This is the rotation angle and represents the width of the projection, so W = 512. Finally, they used 5 features per point in the point cloud to perform their experiments: the (x, y, z) coordinates, an intensity measurement and a depth value, so C = 5.

Predicting a robust and accurate point-wise label map requires understanding of high-level and low-level details. With the use of down-sampling operations, there could be significant loss of low-level semantics, which may in turn, generate inconsistent and spurious predictions. Conditional Random Fields (CRF) have been extensively used along with CNNs for image-based segmentation to refine inferences and restore low-level details [19], [20], [18]. The work in SqueezeSeg follows a similar approach as [22] to refine the label map generated by their CNN. They formulated the CRFs as Recurrent Neural Network layer as in [18], and added an additional module to their pipeline that takes the output of the CNN as a probability map and refines it using Gaussian Kernels that aggregate probabilities of the neighboring points. In our work, we evaluate if the Recurrent CRFs are capable of recovering low-level details from point clouds. This is particularly important in the context of sparse point clouds since we are interested in increasing the amount of contextual information that the model has to make better predictions.

B. Ground Annotation

In our work, we adapt the dataset used in the SqueezeSeg approach. This dataset is a modified version of the KITTI dataset that only considers a front-view of 90° (i.e. in the direction of movement) of the point clouds instead of a full-view of 360°. Furthermore, this dataset only contains labeled examples of cars, pedestrians and cyclists, and the remaining points are labeled as unknown. However, as mentioned before, our interests include performing segmentation of ground points. Nevertheless, manual annotation is a rigorous task. Thus, we designed an automatic ground annotation tool based on an plane-fitting algorithm [8], [4] to automatically label ground points from the KITTI set.

The algorithm discriminates deterministically between ground and non-ground points based on two main assumptions. First, that the points that correspond to the ground have the lowest height. Second, that we can describe the geometry of the ground using the linear equation of a 3D-plane (3), (4).

$$ax + by + cz + d = 0 \tag{3}$$

$$-(\mathbf{n}^{\mathrm{T}}\mathbf{p}) = d \tag{4}$$

where the the orthogonal vector is $\mathbf{n} = [a, b, c]$ and the coordinate point is $\mathbf{p} = [x, y, z]$.

Although, in environments such as those analyzed in this work, the ground usually does not present significant irregularities, it does not form a perfect plane, so it is important to consider changes in the steepness of the ground. To do so, we follow the technique used in [4] in which the point cloud is evenly separated into a number of segments, N_{seg}, along the direction of movement (usually the x-axis), as shown in Algorithm 1. Then, we estimate the ground plane on each of the segments.

The first step to estimate the plane is to extract initial seed points by sorting the point cloud segment along the height axis and taking the first N_{points} the median values of these points. The resulting values are used as a threshold to determine the seed points. In [4] they use the mean values to obtain the seeds, however we compute the medians to avoid being affected by outliers.

After obtaining the seed points we iteratively adjust the plane normal by re-computing the seeds on each iteration. We obtain point p_0 which is the median x, y, z values of the seeds, and use it to estimate the normal n vector of the plane. To do this, we capture the dispersion of the seed points through a covariance matrix and using Singular Value Decomposition (SVD). With both vectors we compute d_{max} which is used as the distance threshold from a point to a plane. Then, for each point in the point cloud segment we compute its orthogonal distance and compare it against the threshold value to determine if the point will be used as a seed. We repeat this process N_{iter} times to refine the plane

Algorithm 1 Ground Labeling Algorithm

Input: P_{in}: input point cloud without ground labels
Output: P_{out}: output point cloud with ground labels

Initialize:
  P_{seg}: input point cloud segment
  N_{points}: number of points used to estimate seeds
  N_{seg}: number of point cloud segments
  N_{iter}: number of plane estimations per segment
  Th_{seed}: seed selection threshold

Begin:
sort(P_{in}, axis=x)
dividePointCloud(P_{in}, N_{seg})
for P_{seg} : P_{in} **do**
  $seeds \leftarrow$extractInitialSeeds(P_{seg}, N_{points}, Th_{seed})
  for i : N_{iter} **do**
    $p_0 \leftarrow$computeMedians(seeds)
    $n \leftarrow$estimateNormal(seeds, p_0)
    $d_{max} \leftarrow$distance(n, p_0)
    seeds.clear()
    for p : P_{seg} **do**
      if $distance(n, p) < d_{max}$ **then**
        seeds.push(p)
  for p : P_{seg} **do**
    if $distance(n, p) < d_{max}$ **and** p.label == UNKNOWN **then**
      $p.label \leftarrow GROUND$
  P_{out}.push(P_{seg})
End

extractInitialSeeds:
sort(P_{seg}, axis=z)
$Th_{lowest} \leftarrow computeMedians($$P_{sort}$(1:$N_{pts}$) + Th_{seed}
for p : P_{seg} **do**
  if $p.height < Th_{lowest}$ **then**
    seeds.push(p)
return seeds

normal. The final linear model is used to assign labels for the ground class to the point cloud.

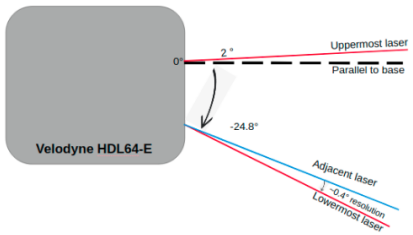
C. Down-sampling

The following step to annotating ground points, is to down-sample the dataset to generate the sparse point clouds required for our experiments. The reason for it, as explained before, is the unavailability of training data and that collecting and annotating data is a labor intensive task. Thus, we propose reducing the vertical density of the original dataset to generate training examples for our experiments.

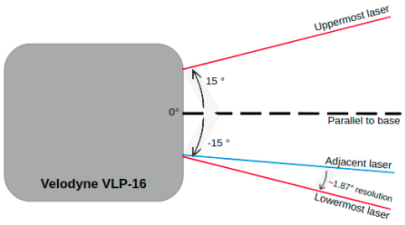
LiDAR point clouds from the KITTI dataset were collected using a Velodyne HDL-64E which has a total of 64 channels with a vertical resolution of approximately 0.42° within a field of view (FOV) of 26.9° (Fig. 2 (a)). However, we are interested in working with sensors that have higher vertical sparsity, hence lower resolution. One such example is the Velodyne VLP-16 which has 16 channels and a vertical resolution of approximately 1.8° to 2° (Fig. 2 (a)).

Note that vertical FOV of Fig. 2 (a) goes from 2° to -24.8°, whereas the FOV of Fig. 2 (b) goes from ±15°. In this work, we assume that we would be able to tilt the Velodyne VLP-16 to align it with the vertical FOV of the Velodyne HDL-64E.

Our procedure to down-sample consists on removing all points from a set of rings in the dense point cloud, depending on the desired resolution of the sparse point cloud. To do so, it is necessary to obtain the ring information per point



(a) Verical field of view of a Velodyne HDL-64E



(b) Vertical field of view of a Velodyne VLP-16

Fig. 2: Vertical FOV of a sensor with high vertical density (a) and a sensor with lower vertical density (b).NOTE: angles are not to scale, they are used for visualization purposes.

measurement as,

$$ring = \frac{\theta_{\max} - \theta}{\Delta\theta} \tag{5}$$

where $\theta_{\max}$ = 2°, is the angle of the uppermost laser beam in the Velodyne HDL-64E, θ is the zenith angle obtained from (1) and the vertical resolution, $\Delta\theta$ = 0.42°.

Here, we illustrate an example to produce a down-sampled point cloud from one that has a high density. To generate scans that resemble those produced by a sensor with low vertical resolution like the Velodyne VLP-16 (Fig. 1 (b)), our desired resolution should be close to the range 1.8° to 2°. Based on the vertical resolution of dense point clouds from KITTI (Fig. 1 (a)), the closest resolutions we can obtain are 1.68° and 2.1°, which are the spanning angle between 4 and 5 consecutive laser beams. This means that to produce the sparse point cloud it would be necessary to keep only the information of either 1 out of 4 rings or 1 out of 5 from the original point cloud.

IV. IMPLEMENTATION DETAILS

A. SqueezeSeg

As explained in (Section II. A), the SqueezeSeg CNN takes an input of H x W x C. In our experiments, the channel size of our target point clouds is 16, so $H = 16$. We keep the width W and the number of features C unchanged. To operate with point clouds with 16 ring we slightly modify the number of filters required by the network modules *fireModules*[21] and *fireDeconvs*. We refer the readers to [9] for further details about the structure of these modules and how to modify them.

B. Ground Annotation

As mentioned in the previous sections, we designed the ground annotation tool to label the dataset used on the SqueezeSeg approach. We used our ground annotator to label a total of 10,278 point clouds, and we empirically set our calibration parameters as follows: $N_{seg} = 4, N_{points} = 20, N_{iter} = 3, Th_{seed} = 0.8$ m.

C. Downsampling

For simplicity and as proof of concept, in our experiments, we reduced the density of the original point clouds from 64 channels to 16 channels to obtain sparse point clouds with a vertical resolution of 1.68°. Fig 1 shows an example of a resulting point cloud after reducing the vertical density from 64 channels to 16 channels.

D. Training

To demonstrate our approach, we present two experiments using three classes *ground*, *car* and *person* which comprises both *pedestrian* and *cyclist* classes. In the first experiment we train the network model without modifications using the original dataset. In the second experiment we train and evaluate the network that operates on sparse point clouds with the dataset that was down-sampled to 16-ring point clouds.

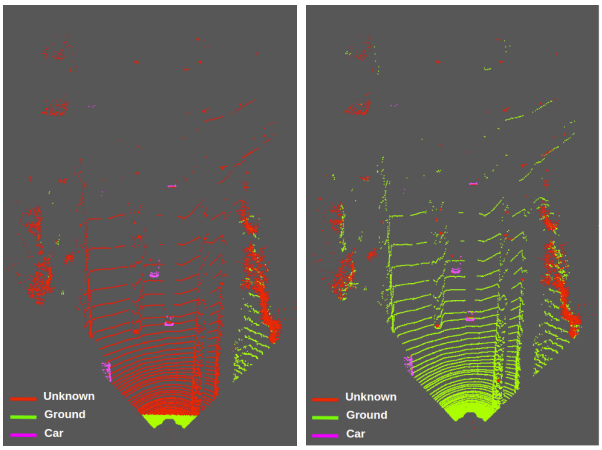
The original dataset, as well as the generated datasets are split into a training set with 8,736 point cloud frames and a validation set of 1,542 frames. We make sure that the sequence frames in the training set do not appear in the sequence frames of the validation set because frames can be temporarily correlated if they are from the same sequence [9]. To perform the training experiments we use an NVIDIA GeForce GTX 1080 Ti, for the evaluation and testing we use an NVIDIA GeForce 1060 Max Q.

For both experiments, we present and compare results when the prediction map of the CNN is fed to the CRF layer against the results when CRFs are not used. This is to evaluate if the Conditional Random Fields proposed by SqueezeSeg will allow to refine the prediction maps of the sparse point clouds. One objective of using CRFs is to restore low-level details [19], [20], [18] that might be lost during pooling operations that reduce the spatial size of the data. Evaluating the performance of CRFs in sparse point clouds is particularly important because due to the nature of the data, it is significantly harder to extract meaningful structural information of object instances and, low-level details are less evident in these point clouds. This loss of object structure can be observed in Fig. 1. Finally, the performance metrics presented on each of our experiments are the precision (P), recall (R) and intersection over union (IoU) obtained during the validation processes.

V. RESULTS

A. Ground Annotation

We compare our implementation with the Ground Plane Fitting (GPF) technique in [4], which was also tested on the KITTI dataset. We observe that in general, both methods achieve similar results. However, GPF significantly underperforms when the seed points used to estimate the 3D plane include outliers (Fig. 3). This is because they compute the mean values of the seed points to obtain the distance threshold to discriminate between ground points and non-ground points. Thus, noisy measurements that have a highly negative height value have a severe impact on the segmentation results. We addressed this situation simply by computing the median values of the seed points to obtain the distance threshold instead of the means values.



(a) GPF result **(b)** Our result

Fig. 3: Example of the impact of outliers in [4] resulting in under-segmentation of ground points (a). Result of our implementation (b).

B. Experimental Results

Following the experimental setup proposed in the SqueezeSeg approach, we summarize accuracy scores of the experiment with the down-sampled dataset in Table I and the original dataset in Table II. In both experiments, we achieve good results for the ground class and in general, high recall scores. The latter is important in applications like autonomous navigation as the cost of missing a positive value might be higher and lead to accidents compared to the cost of including negatives. In both experiments the accuracy scores of the remaining classes is notably reduced. We associate these results to the unbalance of the dataset, i.e. significantly fewer points are attributed to the car instance than to the ground instance.

For the experiment carried out on the sparse point clouds, we observe that using Conditional Random Fields to refine the prediction significantly improved the prediction maps. However, we obtain similar results with and without the CRF layer on the experiment with the dense point clouds. In the original approach, they present a similar behavior in their results. They attribute this situation to the empirical choice of hyperparameters used to compute the Gaussian Kernels in the CRFs.

In Table III, we summarize the average prediction time per point cloud frame of our experiments. Although using CRF notably increases the processing time, the prediction times

	Down-sampled Dataset			
	CRF	P	R	IoU
Ground	w	0.9760	**0.9118**	**0.8905**
	w/o	**0.9762**	0.8898	0.8758
Car	w	**0.4528**	0.9157	**0.4342**
	w/o	0.4027	**0.9158**	0.3883
Person	w	**0.1249**	**0.6697**	**0.1192**
	w/o	0.0579	0.6283	0.0561

TABLE I: Performance on the validation set of the 16-ring down-sampled dataset.

	Original Dataset			
	CRF	P	R	IoU
Ground	w	0.9890	0.9194	0.9101
	w/o	**0.9904**	**0.9212**	**0.9131**
Car	w	**0.6859**	0.9854	**0.6791**
	w/o	0.6850	**0.9857**	0.6783
Person	w	**0.4468**	**0.8701**	**0.4189**
	w/o	0.4354	0.8698	0.4087

TABLE II: Performance on the validation set of the original dataset.

are faster than common sampling rates of LiDAR scanners (10 - 20 Hz). This is particularly important for embedded applications in robotic perception and autonomous vehicles where computation resources are limited.

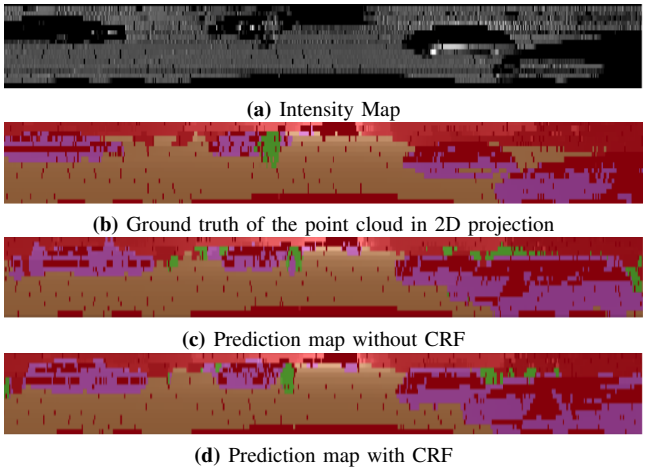
In Fig. 4 and Fig. 5 we compare the prediction results from the original dataset (Fig. 5) and the down-sampled dataset (Fig. 4) represented in the 2D projection. In both figures we compare the same frame from a sequence obtained from the KITTI dataset. We observe that compared against the ground truth (a), both experiments show accurate predictions (b) and (c), and also that using CRF helped refining spurious predictions. However, predictions from the down-sampled dataset in some cases mislabel object instances. One such example can be observed in the visualizations from the experiments, where objects like poles, letter boxes or trash cans are mislabeled as *person*. As discussed before, this might be because there are significantly less training examples for this class compared to *ground* and *car*.

VI. CONCLUSIONS

We explored a method to perform fast and accurate semantic segmentation of highly sparse point clouds in urban environments. In particular, we focused on segmenting

	Average prediction time (ms)	
	Down-sampled Dataset	Original Dataset
w/ CRF	12.36	23.11
w/o CRF	9.42	18.30

TABLE III: Prediction time per frame.



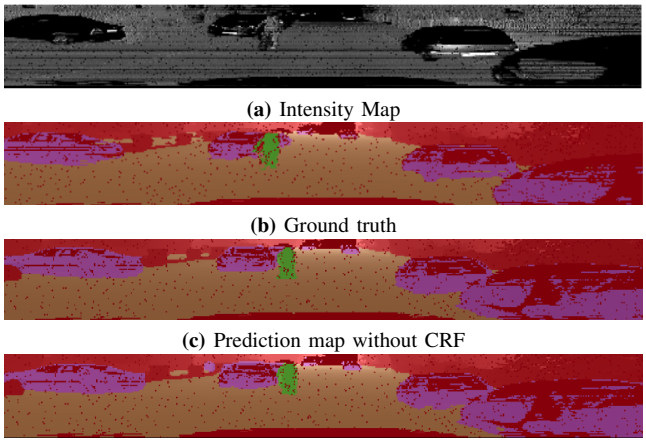
(a) Intensity Map

(b) Ground truth of the point cloud in 2D projection

(c) Prediction map without CRF

(d) Prediction map with CRF

Fig. 4: Visualization of predictions on a 2D projection of a LIDAR point cloud using the down-sampled dataset. Note that these images were expanded along the height for visualization purposes.



(a) Intensity Map

(b) Ground truth

(c) Prediction map without CRF

(d) Prediction map with CRF

Fig. 5: Visualization of predictions on a 2D projection of a LIDAR point cloud using the original dataset.

person, *car* and *ground* instances. Using Deep Learning allowed us to address some of the difficulties presented in previous work on these type of point clouds by taking advantage of a CNN that does not rely on hand-crafted features, nor iterative algorithms that are subject to instability or are not scalable to unstructured environments. As a proof-of-concept, we designed a simple method to obtain training data that resemble the data produced by our target sensor, i.e. Velodyne VLP-16. We also extended the CNN capabilities to segment ground. In general, our results show very high accuracy on ground detection and general robustness against noise. We mainly attribute lower scores for the *car* class to the unbalance of classes.

In future work, we foresee several pathways. First, there is potential in exploring semantic segmentation in unstructured environments. Then, we intend to add the *pedestrian* and *cyclist* instances to the system, instead of only the *person* class. There is also room for improvement in regard of making a more hierarchical description of the ground, e.g. detecting

sidewalks and intersections. Finally, we aim to integrate this system on an NVIDIA Jetson TX2 as a potential application in mobile robot perception in dynamic environments.

ACKNOWLEDGMENT

This work was supported by the Instituto Tecnológico y de Estudios Superiores de Monterrey and by The Robotics Institute through the RISS program.

REFERENCES

[1] M. Li and D. Yin, "A fast segmentation method of sparse point clouds," in *2017 29th Chinese Control And Decision Conference (CCDC)*, pp. 3561–3565, May 2017.

[2] K. Midlicki, M. Pajor, and M. Sakw, "Real-time ground filtration method for a loader crane environment monitoring system using sparse lidar data," in *2017 IEEE International Conference on Innovations in Intelligent SysTems and Applications (INISTA)*, pp. 207–212, July 2017.

[3] D. Z. Wang, I. Posner, and P. Newman, "What could move? finding cars, pedestrians and bicyclists in 3d laser data," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4038–4044, May 2012.

[4] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5067–5073, May 2017.

[5] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion," in *2009 IEEE Intelligent Vehicles Symposium*, pp. 215–220, June 2009.

[6] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3d lidar point clouds," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2798–2805, May 2011.

[7] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *International Conference on 3D Vision (3DV)*, 2017.

[8] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for very fast ground segmentation in velodyne lidar data," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 97–103, April 2018.

[9] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," *arXiv preprint arXiv:1710.07368*, 2017.

[10] M. Simon, S. Milz, K. Amende, and H. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," *arXiv preprint arXiv:1803.06199*, 2018.

[11] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 1696–1704, Curran Associates, Inc., 2016.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," 2013.

[13] M. Himmelsbach, T. Luettel, and H. J. Wuensche, "Real-time object classification in 3d point clouds using point feature histograms," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 994–1000, Oct 2009.

[14] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 163–169, Oct 2016.

[15] Q. Zhang, X. Song, X. Shao, H. Zhao, and R. Shibasaki, "Unsupervised 3d category discovery and point labeling from a large urban environment," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2685–2692, May 2013.

[16] C. Qi, H. Su, K. Mo, and L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *arXiv preprint arXiv:1612.00593*, 2016.

[17] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*, 2017.

[18] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and S. Torr, "Conditional random fields as recurrent neural networks," in *International Conference on Computer Vision (ICCV)*, 2015.

[19] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *CVPR*, 2016.

[20] P. Krhenbhl and V. Koltune, "Efficient inference in fully connected crfs with gaussian edge potentials," in *NIPS*, 2016.

[21] F. Iandola, H. Song, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[22] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," in *arXiv*, 2016.

Toward Robust Stair Climbing of the Quadruped using Proprioceptive Sensing

Zhiyi Ren[1] and Aaron Johnson[1]

***Abstract*—Uneven stairs and ground obstacles often cause legged robots to trip and even fall over. Autonomous traversal requires robots to detect leg disturbances accurately and react in time. We present an approach that combines a leg observer for contact detection with reactive behavior for recovery. Exploiting the mechanical transparency of the robot design, the observer achieves leg proprioception and contact identification with minimal delay. Reactive algorithms then trigger the robot to switch between walking and stair-climbing, or to adjust swing leg trajectories to step over obstacles. We demonstrate the reliability and potential of the contact observer on robust stair climbing. We envision that future work will establish an autonomy framework for legged robots to traverse through multi-component natural terrains using similar proprioceptive sensing strategies and reactive behaviors.**

I. Introduction

Legged robot can traverse through various difficult terrains that obstruct traditional wheeled or tracked robots, benefiting from large leg workspace and contact geometry reconfiguration [1]. However, as one of the most common obstacles in both indoor and outdoor environments, stairs remain challenging for legged robots to climb over. Due to the inherent geometry of normal stairs, small-sized legged robots often need to use more dynamic gaits such as jumping and bounding [2], and compositions of these dynamic behaviors often risk the stability of the robots. Timely transitions between behaviors require precise detection of the stair tread. In addition, any unexpected disturbances on the stairs may cause the robots to lose balance and even fall over, demanding quick detection and recovery.

For robust autonomous stair climbing, we believe that exteroception with camera and other sensors is essential to the overall system in the future. But robots cannot entirely rely on the the visual approach: small bumps on the stairs may be difficult for the camera to notice, and the camera view is often occluded. Computer vision algorithms are yet to be developed to accommodate the highly dynamic motion of legged robots, with the state of the art likely to generate insufficiently precise stair location data. Meanwhile, proprioceptive sensing provides a more intuitive and reliable solution to contact and disturbance detection. Using motor encoders and IMU built-in on the robot, it can "feel" the object and perceive its orientation as it walks through terrains. It is usually computationally inexpensive, thus reducing the detection latency and allowing for in-time recovery.

Our vision is to let the robot traverse through any type of stairs in any environment. Most of the existing stairs are designed specifically for human walking, and the relative size of the stair may pose a problem for the robot. Our earlier attempts showed that the high slope of some stairs consistently led to loss of stability during behavior transitions. Furthermore, outdoor stairs are often rugged and uneven, even sometimes tripping humans. To simulate a similar setting in an indoor, controlled environment, we chose a set of stairs with a smaller slope (about 30 degrees), and place obstacles on the stairs. The robot is expected to trot towards the stair, detect and ascend, and recover from disturbances.

Fig. 1. Minitaur robot on the stairs used for testing.

Here we present the beginning of a solution to robust stair climbing of the robot. The gaits used include trotting on the ground for approaching the stairs, trotting on the stairs for aligning with the next one, and bounding of the front and rear legs alternatively for ascending the stairs. Leg state estimation is achieved with both offline simulation and online observers, which only require motor shaft encoders as the onboard sensor. As the robot walks towards the stair with a pre-defined leg trajectory, the actual leg positions from the encoders are compared to the outputs of the offline simulation, forming the residual values. A significant residual value during the expected air phase indicates the leg contact with the first stair, and signals the robot to start bounding up. Meanwhile, upward bounding of the legs does not follow a specific leg trajectory, and a online observer is necessary to generate expected leg states. High residual values indicate either the leg touchdown or the presence of obstacles at the legs, which quickly adapts to a certain trajectory for recovery.

The paper is organized as follow: the following subsection

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213. zhiyi.ren@jhu.edu, amj1@cmu.edu

presents major related works in robot stair climbing, and some background and justifications of the robot used in our studies. Section II introduces the proprioceptive sensing in more details, explaining the offline simulation and online observer in the use of disturbance detection. Section III demonstrates the gait compositions for stair climbing, and the reactive behaviors for disturbance recovery. Section IV documents the experimental results of the detection algorithm and reactive behavior. The paper finishes with a conclusion and multiple future directions.

A. Related work

Stair climbing has been researched with different types of robots and strategies. Tracked robots enjoy enhanced stability over traditional wheel robots thanks to greater ground contact area and lower center of gravity. Xiong and Matthies [3] demonstrated using computer vision algorithms to locate the stair edges and determine the relative orientation of the stair and robot for guidance. Steplight et al. [4] developed a hierarchical stair-climbing model using sensor fusion for a tracked robot, and work in [5] improved the robustness of the sensor fusion approach by introducing extended Kalman filtering for state estimation. Meanwhile, few legged robots have been used in stair climbing. One of the earliest versions of the RHex hexapod robot exploits a special curved leg design [6] for stair clmbing. The RHex robot further incorporated stair sensing and sequential compositions of controllers to achieve autonomy [7], [8].

However, almost all of these works demonstrated the climbing strategies on regular indoor stairs with no obstacles. Some of the works [5] dealt with collisions with stairs but not any unexpected irregularities of the stair shape. It is certain that most of the algorithms would fail in an outdoor rugged environment, where the stairs may be littered with small obstacles.

Fig. 2. Outdoor stairs at Schenley Park, Pittsburgh, PA.

We addressed the problem with a quadrupedal platform, the Minitaur from Ghost Robotics (Fig. 1). Each leg is lightweight (Table 1) and driven by two DC brushless motor with no gearbox. This direct-drive design results in high mechanical transparency [9], which enables high leg acceleration and ground impulse detection in minimal delay. It has been demonstrated that the Minitaur is capable of a series of stable, dynamic maneuvers including bounding and trotting [10]. We believe that the high maneuverability and high mechanical transparency render the robot a competent choice for stair climbing. It is also equipped with motor encoders and an IMU, and the STM32 ARM microcontroller onboard updates motor commands at a maximum of 1 KHz subject to the amount of computations.

One issue that we encountered was that the leg length of Minitaur is almost the same as the height of normal-sized stairs, which causes difficulty in walking up the stairs. Topping et al. [2] documented the quasi-static mismatch between Minitaur and normal-sized stairs due to the leg size and torque limit, introducing a dynamic, pronking-like gait for stair ascent. However, we believe that this gait is vulnerable to any disturbance on the stairs. With all four legs in the air and hence no ground contact when any collision occurs, the robot is likely to lose stability and fall over on the stairs. Instead we adopt a less dynamic, bounding-like gait, with front and rear pairs of legs ascending alternatively. While the other pair of legs is anchored to the floor, the legs in the air are able to quickly adjust the trajectory when contacting obstacles, maintaining whole-body stability.

While we intend to develop a complete stair climbing behavior for the Minitaur, our focus in this paper, as the first step towards robust traversal over uneven stairs, is to introduce proprioceptive sensing for obstacle detection. Disturbance observer has been applied on multiple legged robot platforms, including the RHex robot [11] and some humanoids [12]. While many other novel approaches, such as using particle filtering [13] or probabilistic contact fusion [14] have emerged, we believe that a simple, deterministic observer model is sufficient to spot the changes in leg dynamics caused by the obstacles. Also, the simple model is less demanding in computations for the sole onboard microcontroller. As the target terrain becomes more complicated, we would improve upon the simple model in the future.

II. Proprioceptive Sensing

The general approach is to simulate the 2D leg motion in finite time steps. We ignore the contact force and focus on the period when the legs are in the air. Significant deviations from the expected states (high residuals) indicate that the leg has "felt" a stair or obstacle.

A. Leg Dynamics

The Minitaur robot has a symmetric five-bar mechanism for each of the four legs. Each leg is driven by two brushless DC motors to move in the sagittal plane. The kinematics of the mechanism is detailed in [9]. We assume that the center of mass of each link is located at the geometric center.

We use Euler-Lagrange equations to solve the leg dynamics. The generalized coordinates are $\theta \in \mathbb{R}^{2\times 1}$, the two motor angles. Consider the following equation of motion.

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) = \tau \quad (1)$$

where $M \in \mathbb{R}^{2\times2}$ is the mass matrix obtained by combining the mass each of the four link in a leg, $C \in \mathbb{R}^{2\times2}$ is the Coriolis matrix, $N \in \mathbb{R}^{2\times1}$ is a vector that contains gravity and joint friction terms, and $\tau \in \mathbb{R}^{2\times1}$ is the torque vector from the ideal motor model based on the applied PWM,

$$\tau = K_t I \quad (2)$$

$$I = \frac{0.95 \cdot V \cdot PWM - V_{emf}}{R} \quad (3)$$

$$V_{emf} = K_t\dot{\theta} \quad (4)$$

where K_t is the torque constant (and the back electromotive force (emf) constant), V is the supplied voltage measured by onboard sensor, R is the armature resistance. The 5% voltage deduction is built-in and confirmed with Ghost Robotics. A standard proportional-derivative (PD) controller determines the motor PWM value as follows,

$$PWM = K_p(\theta_d - \theta) + K_d(0 - \dot{\theta}) \quad (5)$$

where K_p and K_d are the proportional and derivative gains, and θ_d is the target position. Note that we do not set a reference velocity here. The controller gains are tuned for stable behavior.

B. Model Parameters

Accurate model parameters are critical to the performance of the offline simulation and online observer. Below are all the major parameters that we considered.

TABLE I
MODEL PARAMETERS, FINAL VALUES, AND SOURCES

Limb		
Parameters	Value	Source
Upper limb length (m)	0.1	Measured
Lower limb length (m)	0.2	Measured
Upper limb mass (kg)	0.040	Measured
Lower limb mass (kg)	0.081	Measured
Upper limb inertia (kg-m^2)	0.00023	Estimated
Lower limb inertia (kg-m^2)	0.00068	Estimated
Motor		
Torque Constant (Nm/A)	0.0959	Estimated
Rotor Inertia (kg-m^2)	0.00005	Estimated
Static Friction (Nm)	0.056	[9]
Kinetic Friction (Nm)	0.023	[9]
Viscous Friction ($\frac{Nm}{rad/s}$)	0.00013	[9]
Armature Resistance (Ω)	0.180	Estimated
Others		
Joint Viscous Friction ($\frac{Nm}{rad/s}$)	0.0037	Estimated
Time delay (ms)	3	Estimated

1) Limbs: Limb length and mass are measured directly. Initially we estimated the limb inertia from the CAD model, but the extra weight of the bolts and bearings at the joints were ignored in that case.

2) Motors: The motor friction values are from [9], which also provided an overestimated value of the motor rotor inertia. The torque constant and armature resistance values are both provided by the motor manufacturer, but any partial burnout of the motors may have affected the values.

3) Time delay: For the offline model, the time delay between the mainboard and the motor controller needs to be taken into account. The effect of the time delay on the stability of the Minitaur robot is documented in [15]. When we ran simulation with a fairly long time delay (10ms), the result showed instability of the legs.

Overall, we estimated the limb inertia, torque constant, armature resistance, rotor inertia of the motor, and time delay. We collected the sample data by running the triangular trajectory of the legs in the free air, and the speed ramped up from 1 to 2.5 strides per second. The cost function was set to the total error per millisecond (per update) from all the samples. We chose the constrained Nelder-Mead method [16], which uses variable transformation to set the upper and lower bounds of the estimated parameters. We ran the method with different initial values within the bounds. The results were consistent and reasonable, listed in the Table 1. The estimated limb inertia is higher than the estimations from the CAD model, possibly due to the extra weight of bolts and bearings. The torque constant and armature resistance provided in the motor specifications is 0.0959 Nm/A and 0.186 Ω respectively, exactly the same as or very close to the estimation results. The estimated time delay value is consistent with [15].

C. Offline Simulation

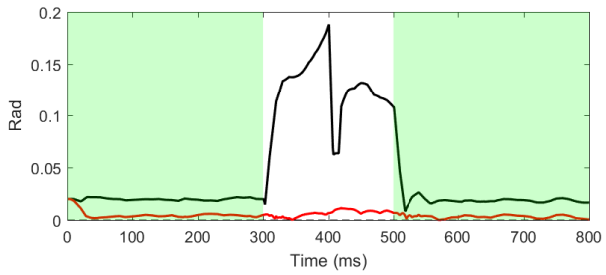
When the robot trots on the ground, the legs follow a pre-defined reference trajectory, an isosceles triangle in the Cartesian space. We simulated the trotting gait of the robot with 1ms time step in Matlab by solving the leg dynamics. The outputs of the simulation, $\hat{\theta}$ and $\hat{\dot{\theta}}$, are compared to the leg position θ and velocity $\dot{\theta}$ recorded in real robot trials, and the observer residuals of the simulation are formed by taking the average of the absolute values of both motors:

$$r_\theta = \frac{sum(|\hat{\theta} - \theta|)}{2} \quad (6)$$

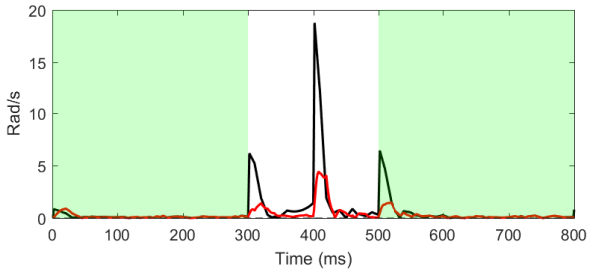
$$r_{\dot{\theta}} = \frac{sum(|\hat{\dot{\theta}} - \dot{\theta}|)}{2} \quad (7)$$

We collected leg data at a median speed (1.25 stride per second) in both conditions of swinging in the air and trotting on the ground, and plot r_θ and $r_{\dot{\theta}}$ with controller tracking error ($\theta_d - \theta$ and $\dot{\theta}_d - \dot{\theta}$) in Fig. 3 and 4. The shaded green region indicates the expected stance phase of the gait.

The figures show that the tracking errors in the expected flight phase is much higher than the observer residuals. Due to the nature of PD controller, the leg cannot follow the corners of the triangular trajectory well where high acceleration occurs, while the model-based simulation accounts for the inertial effect. Therefore, if the leg experiences any disturbance in the air, the increasing residual values should indicate the event,

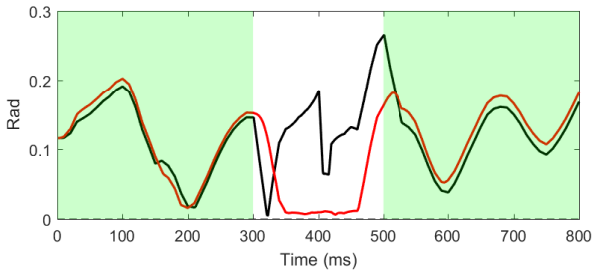


(a) Position residual (red), r_θ, and position tracking error (black), $\theta_d - \theta$

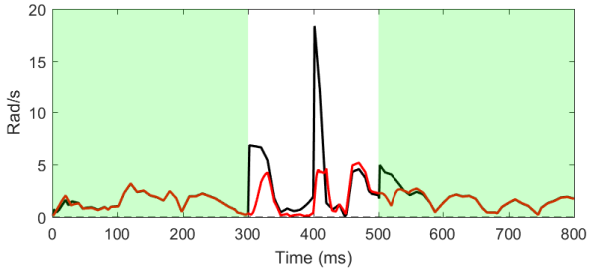


(b) Velocity residual (red), $r_{\dot{\theta}}$, and velocity tracking error (black), $\dot{\theta_d} - \dot{\theta}$

Fig. 3. Observer residual compared to PD controller tracking error in position and velocity in a free swinging cycle (no disturbance from the ground). Green shading indicates the expected stance phase.



(a) Position residual (red), r_θ, and position tracking error (black), $\theta_d - \theta$



(b) Velocity residual (red), $r_{\dot{\theta}}$, and velocity tracking error (black), $\dot{\theta_d} - \dot{\theta}$

Fig. 4. Observer residual compared to PD controller tracking error in position and velocity in a trotting stride cycle (disturbance from the ground). Green shading indicates the expected stance phase.

while the tracking errors would not be able to. Although the velocity residual values show the similar trend, the position data alone was sufficient to capture the abrupt change in leg dynamics.

Note that during the expected stance phase, both residual and tracking error values are high since we ignore the ground contact force in the leg dynamics. The leg disturbance during the stance phase is not within the interest of this paper, and would require contact force estimation.

D. Online Observer

While the robot trots on the flat ground with a pre-defined trajectory, the legs do not follow a specific trajectory when the robot climbs up stairs. Many of the dynamic maneuvers on the stairs are closely sequenced and some involve open-loop power. While this can still be simulated, the errors in residual values are likely to accumulate, and affect the initial conditions of the legs at a certain step within the sequenced behavior. This leads to the necessity of a memoryless, online observer that predicts leg states as the robot bounds up stairs. At each time step, the observer reads the current states of the legs from the motor encoders, and outputs the predicted states at the next time step for comparison.

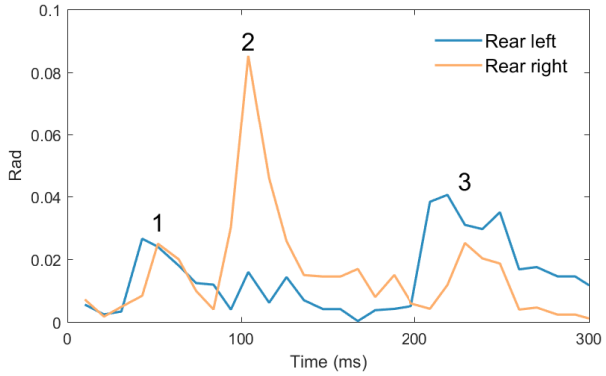
1) Reduced dynamics: Since the computational power of the onboard microcontroller is limited, we adopted a reduced dynamics model instead. The Coriolis terms take a fair amount of overall computations, but the values are usually small enough to be negligible. We also remove the joint friction terms. The equation of motion is reduced to:

$$M(\theta)\ddot{\theta} + N(\theta) = \tau \tag{8}$$

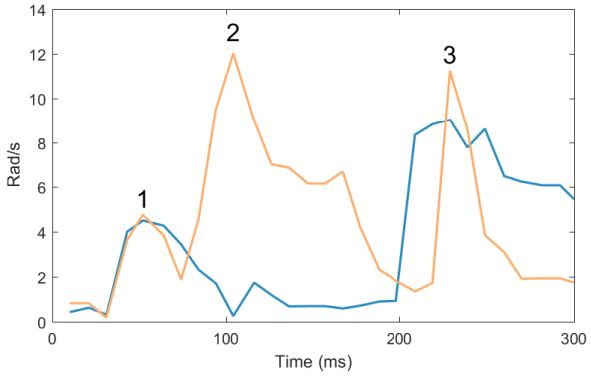
Since the robot always has at most two legs in the air when it climbs up stairs, only two observers run at the same time. The computations benchmark at about 2ms per cycle. However, the leg states do not change significantly in 2ms even if the leg hits an obstacle. When the rear legs lift off the ground and swing forward in the air, the residual shows a small amount of error due to the inertial effect, which is comparable to the residual caused by the obstacle. Therefore, we extended the observer time step from 2ms to 5ms, and the results were able to differentiate hitting the obstacle from swinging forward as shown in Fig. 5. Now the motor commands update at 200 Hz during stair ascent as opposed to 1 KHz during trotting. The update rate is still fast enough for quick reactive behaviors in the case of leg disturbances.

Given the changes in the model, we re-ran the parameter estimation using the same constrained Nelder-Mead method on the same selected parameters, except for the removed joint viscous friction coefficient. The results were very close to those listed in Table 1 and the differences are negligible.

To compare the performance of the reduced and full models, we calculated the position residuals summed in a free swinging cycle at a median speed (1.25 strides per second) using both models. We also simulated the 200 Hz update rate on both models. The results are listed in Table 2 below. The lower update rate does not affect the performance significantly. The model reduction results in an increase in errors, but still within a reasonable range. The goal of the online observer is to capture the sudden impulse at the leg instead of tracking a specific trajectory, thus not as demanding in performance. Fig. 5 shows that the reduced model is sufficient to pick up such impulses from obstacle contacts.



(a) Position residual



(b) Velocity residual

Fig. 5. Online observer residuals when the rear legs are in the air and the rear right leg hits an obstacle. The three peaks corresponds to: 1. swinging forward, 2. obstacle contact, 3. touchdown

TABLE II
PERFORMANCE OF THE FULL AND REDUCED MODELS AT DIFFERENT UPDATE RATED

Model Type	Update Rate (Hz)	Error (Rad)
Full	1000	7.57
Full	200	7.68
Reduced	1000	16.60
Reduced	200	16.86

III. STAIR CLIMBING AND REACTIVE BEHAVIOR

After we developed both the offline model-based simulation and the online observer, we would like to apply these strategies in rugged stair climbing. However, we quickly encountered difficulty in developing a reliable stair climbing gait for the Minitaur. Since the robot is not capable of walking up stairs, it has to use gaits that are more dynamic such as bounding. With no existing controller for stair ascent, tuning open-loop power and controller gains turned out to be very time-consuming. Another issue was that the supporting legs on the ground often slip off the stairs when the other pair bound, causing the robot to fall off the stairs. We reduced the slippage by applying open-loop power to generate more normal force. The current gait is robust enough to ensure that the robot does not slip off the stairs.

The stair climbing behavior consists of a few major components.

A. Trotting on the Floor and Stair Detection

The robot trots with a pre-defined triangular trajectory towards the first stair, and calculates the residual values using offline simulation outputs and motor encoder readings. The robot would check the residual values within a time range when the legs are expected in the air, determined based on offline simulation results.

Initially we would like to put some obstacles on the ground, and the robot would detect them and step over them with fast leg circulation. However, we could not find a method that consistently differentiates the obstacle and the stairs. Therefore the robot now treats the first stair as the disturbance and then starts stair climbing.

B. Legs Bounding

In order to climb the stairs, the front and rear legs of the robot bound upstairs alternatively. The bounding legs push off the ground, retract until minimum length, swing forward in the air, and touch down (Fig. 6). The supporting legs apply open-loop power into stair to generate normal force for balance. Meanwhile, two observers update the residual values of the bounding legs. While the leg touchdown generates high residual values, the expected touchdown time is known to the robot. The robot would treat the high position and velocity residual values as hitting an obstacle if it occurs before the expected touchdown.

C. Disturbance Recovery

Once the leg observer reports the presence of an obstacle, the affected leg retracts to the minimum length and swings forward, stepping onto the obstacle. Meanwhile, the two legs on the other diagonal extend more to lift up the whole robot body.

D. Trot on the stair tread

When both the front and rear legs have bounded up, there is usually some space between the front legs and the next stair. The front legs have to start bounding right next to the stair, otherwise they will not be able to catch it due to geometric limit. We rotate the triangular trajectory of the ground trotting to match the slope of the stairs, and the robot was able to trot forward with small steps and align itself with the next stair. After a certain time threshold, the next bounding cycle starts.

E. Exit the stairs

Detecting the end of staircase is straightforward. Once the IMU reports a very small pitch angle after a bounding cycle, the robot knows that it has reached the top of the stairs. It would trot forward for some distance and stop.

Fig. 6. The rear legs bound up the stair in 450ms.

IV. Experimental Results

We test the performance of the offline simulation, online observer, and stair-climbing gait separately. Eventually we would like to place random obstacles on the stairs and conduct an experiment survey on stairs of various dimensions and surface conditions.

1) Stair Detection with Offline Simulation: We let the robot start trotting at ten locations, ranging from 0.2 meter to 2 meter from the stairs. The robot was able to detect the stairs every time within two leg cycles once it contacted the stair.

Ideally the robot should detect the stair within the same leg cycle. However, we found that the stair contact sometimes occurred right before the leg hit the ground. If the end of the detection time range is set too close to the start of the ground phase, the robot often mistook the ground for the stair. Some delay on the ground is sacrificed for the extra accuracy of stair detection.

2) Obstacle Detection with Online Observer: A brick-like obstacle was placed on the stair, and tripped the rear right leg of the robot as it swung forward in the air. During initial tests, the robot was able to detect the obstacle consistently.

3) Stair Climbing Robustness: We challenged the robot to keep climbing the stairs before it failed. During initial tests, the robot was able to climb about five to ten stairs.

We realized that PWM control for bounding is not sufficient for robustness. As the robot climbed upstairs, the bounding height of the legs decreased given the same PWM values since the motors started to overheat and the battery charge is reduced. The bounding discrepancies caused the robot to trip on the stairs, or steer sideways and lose balance.

V. Conclusion

This work addresses the robustness of the stair climbing of a quadruped by incorporating proprioceptive sensing and climbing behavior. Offline simulation models and online observers are developed for disturbance detection. The robot is able to detect the stairs, bound up the stairs, and react to obstacles by quickly adjusting the leg trajectory. More experiments are yet to be completed. We believe that proprioceptive contact detection and reactive behavior would be crucial to the robust traversal of stairs.

Certainly the overall stair climbing framework is not quite complete yet. Parts of the climbing behavior we introduced rely on open-loop control, causing issues such as slipping of the supporting legs. Although our main focus has been using proprioceptive sensing to improve the robustness, we hope to establish a better stair climbing model for behavioral development, integrating force and impedance control. Another ongoing project at our lab is to investigate the effect of tails on dynamic maneuverability of the Minitaur robot. We believe that active actuation of an inertial or aerodynamic tail would maintain the stability of the robot during stair ascent, and even help achieve certain dynamic behavior that is otherwise infeasible.

Other future works include adding an extra onboard computer to enable online, full dynamics disturbance detection. The extra computational power also enables many other improvements, including whole body modeling, contact force sensing, sensor fusion, and sequential compositions [17] of online controllers. A newer version of the Minitaur robot also provides current sensing, which would allow motor current control and improve the accuracy of the online observer by skirting the ideal motor model.

Beyond the scope of stair climbing, we envision that proprioceptive sensing and model-based behavioral development would be critical components of the overall terrain locomotive framework of the legged robot. The robot may combine these components with other approaches such as vision detection to fulfill robust traversal through any terrains.

Acknowledgment

This work was supported by Robotics Institute Summer Scholar program and Robomechanics Lab at Carnegie Mellon University, PA. We would like to thank Joseph Norby, Nathan Kong, and Barrett Werner for the assistance and helpful discussions.

REFERENCES

[1] S. Kim and P. Wensing, "Design of dynamic legged robots," vol. 5, pp. 117–190, 01 2017.

[2] T. T. Topping, G. Kenneally, and D. E. Koditschek, "Quasi-static and dynamic mismatch for door opening and stair climbing with a legged robot," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1080–1087.

[3] Y. Xiong and L. Matthies, "Vision-guided autonomous stair climbing," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, April 2000, pp. 1842–1847 vol.2.

[4] S. Steplight, G. Egnal, S. . Jung, D. B. Walker, C. J. Taylor, and J. P. Ostrowski, "A mode-based sensor fusion approach to robotic stair-climbing," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 2, Oct 2000, pp. 1113–1118 vol.2.

[5] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, D. M. Helmick, and L. Matthies, "Autonomous stair climbing for tracked vehicles," *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 737–758, 2007.

[6] E. Z. Moore, D. Campbell, F. Grimminger, and M. Buehler, "Reliable stair climbing in the simple hexapod 'rhex'," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 3, May 2002, pp. 2222–2227 vol.3.

[7] G. J. Wenger, A. M. Johnson, C. J. Taylor, and D. E. Koditschek, "Semi-autonomous exploration of multi-floor buildings with a legged robot," in *Unmanned Systems Technology XVII*, vol. 9468. Baltimore, MD: SPIE, April 2015, pp. 94 680B–8.

[8] A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek, "Autonomous legged hill and stairwell ascent," in *Proceedings of the IEEE Intl. Workshop on Safety, Security, & Rescue Robotics*, Kyoto, Japan, November 2011, pp. 134–142.

[9] G. Kenneally, A. De, and D. E. Koditschek, "Design principles for a family of direct-drive legged robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, July 2016.

[10] A. De and D. E. Koditschek, "Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking, and trotting," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 743–778, 2018.

[11] A. M. Johnson, G. C. Haynes, and D. E. Koditschek, "Disturbance detection, identification, and recovery by gait transition in legged robots," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 5347–5353.

[12] K. Kaneko, F. Kanehiro, M. Morisawa, E. Yoshida, and J. Laumond, "Disturbance observer that estimates external force acting on humanoid robots," in *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, March 2012, pp. 1–6.

[13] L. Manuelli and R. Tedrake, "Localizing external contact using proprioceptive sensors: The contact particle filter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5062–5069.

[14] G. Bledt, P. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, to appear.

[15] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," *ArXiv e-prints*, Apr. 2018.

[16] J. D'Errico, "fminsearchbnd, fminsearchcon - file exchange - matlab central." [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon

[17] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

Using Convolutional Neural Networks on Optical Flow for Visual Object Tracking

Vivek Roy[1] David Russell[2], Satyaki Chakraborty[3] and Martial Hebert[3]

***Abstract*— Visual object tracking is the process of following an object through a video, given the position of the object in the first frame of the video. It has applications across robotics, surveillance, and autonomous driving. While enormous progress has been made in recent years, challenging benchmarks such as the Visual Object Tracking (VOT) Challenge [1] have demonstrated that there is still significant room for improvement. We propose a convolutional neural network which learns to track objects using the optical flow between consecutive video frames. This approach leverages the explicit geometric information contained in optical flow, which many other methods may abstract away. We present preliminary results on the VOT dataset and discuss avenues for improvement.**

I. Introduction

Visual object tracking is the process of predicting the location of an arbitrary object through consecutive frames of a video, given only its position in the first frame. Tracking systems are utilized in many domains including robotics, surveillance and autonomous vehicles, as they allow intelligent systems to reason about their surroundings at a higher level.

Tracking can be challenging because in addition to moving, the object may exhibit changes in viewpoint or illumination, undergo occlusion, or other be subject to other variations. While enormous progress has been made, challenging datasets such as VOT [1], demonstrate that many trackers are still poor at handling these challenging situations. Given this, and some speed constraints, trackers are still not able to be deployed in many of the situations where they would be useful.

A common tracking method is constructing correlation filters based on the appearance of the tracked object using these to search the next frame. Since these filters are generated online, using solely from the data at runtime, the miss out on the vast annotated datasets which could be used to learn general relationships offline. For this reason, convolutional neural networks (CNNs) are becoming common components of trackers, as the training process allows them to exploit this data. Many CNN-based networks, such as Siamese Network [2] and GOTURN [3], are able to predict the object's motion between a pair of frames, without needing to utilize the entire history.

[1]Vivek Roy is with the Computer Science and Engineering Department, Jadavpur University vivek at vivekroy.com

[2]David Russell is with the Department of Computer Science, Clarkson University russeldj at clarkson.edu

[1,2]Work done while interning at Carnegie Mellon University, through the Robotics Institute Summer Scholars Program

[3]Satyaki Chakrabory and Dr. Martial Hebert are with the Robotics Institute, Carnegie Mellon University

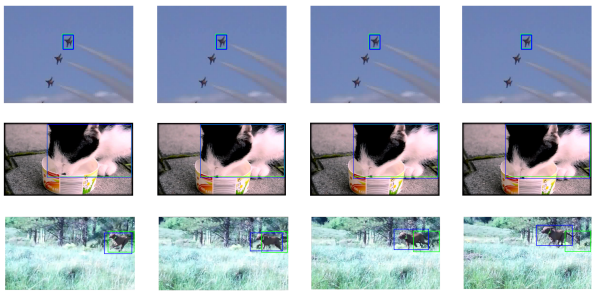


Fig. 1: Results on the validation set of ImageNetVID. Blue boxes represent the ground truth and the green boxes represent the predictions from the proposed approach.

In this work, we build off of approaches like [2] and [3] to create a CNN-based tracker which predicts the object motion between pairs of frames. Our primary contribution is exploring the use of optical flow as an input to the network, rather than image pairs as is currently standard. This optical flow input is computed between the two images the network would normally receive and represents the motion of feature points between frames. This means that the network does not have to learn how to perform feature matching, and can focus on learning more abstract and interesting features of the data. After computing the optical flow for the entire image, we crop it to the location of object in the first frame and resize this crop to a fixed size. This is then fed into our network, which predicts how much the object has moved between the current frame and the next one.

We present preliminary qualitative results of this tracker on a subset of the VOT dataset [1]. Additionally, we propose several improvements to our algorithm, which we believe could improve its performance.

II. Related Work

A traditional approach for object tracking involves building correlation filters which can used to search for the object in the next frame [4]. In general, these filters are trained online, by taking the past crops from the area not predicted to be part of the object and using them as negative samples, while taking the predicted crop as a positive sample. The filter is updated at a certain interval so such that it has a low response for negative samples and a high response for positive ones.This means that as the object is tracked, the network learns an appearance model which incorporates all the different views which have been seen over time. In addition to not being able to utilize the large amount of

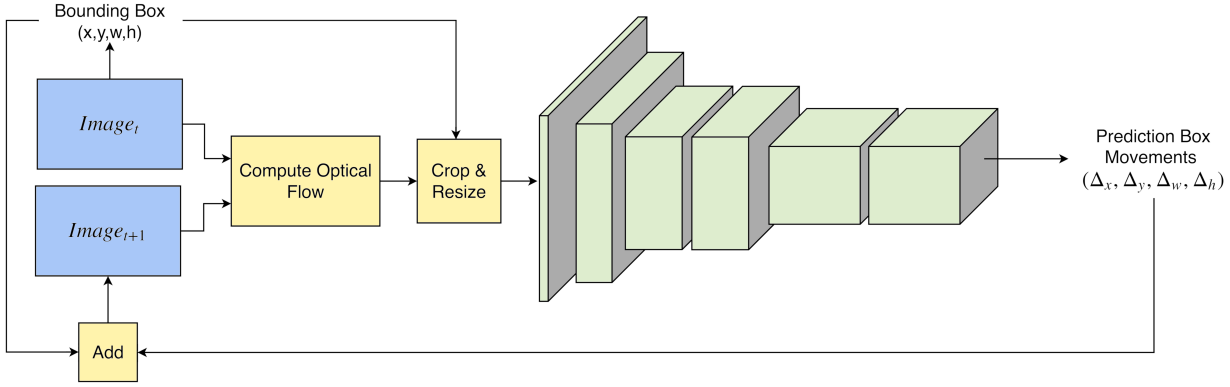


Fig. 2: A high-level diagram of our system architecture.

offline data, these approaches are computationally intensive and far from realtime, because they are training the filters online.

Following the trend in the larger computer vision community, deep convolutional architectures are becoming increasingly common. GOTURN [3] takes a common approach of cropping the current and next frame using the current bounding box, so only the local area affects the prediction. GOTURN uses convolutional layers for feature extraction, and then fully connected layers to reason on these features and produce the motion estimate. In some approaches, notably Siamese Network [2], convolutional layers are used simply to perform feature extraction on the two images cropped images. Then correlation is performed in feature space, and the highest response indicates the expected object position. While successful, this approach looses some low-level features which could be useful. Detect to Track [5], adds one more operation, by computing correlation maps on convolutional features, and then feeding them into a fully connected network. The approaches of [2] and [5] suggests that non-learned operations can be effectively combined with learned ones, and provide motivation for our work using optical flow.

Optical flow, or the estimated displacement between corresponding points in two images, can give us rich information about the geometric structure of a scene. The computation of optical flow has been a topic of interest in the vision community a significant period of time and approaches such as Brox [6] and FlowNet [7] provide compelling results; they are surprisingly robust at estimating feature motion in a variety of challenging situations. This is partly because the the flow is computed using a hierarchical approach which preserves some level of geometric consistency. In fact, optical flow has been used for the basis of geometric trackers such as Median Flow [8]. In this work they simply estimate the robustness of the flow estimation at each point, and then take an average of the displacement suggested by the points which are expected to be most robust. This approach considers all of the points within the rectangular bounding box, whether or not they are likely to lie on the object, and also entirely ignores the points which are not considered to be the best. Our goal with this work is to learn a more sophisticated mapping from optical flow to displacement than can be generated with a hand-crafted approach such as this.

III. Methods

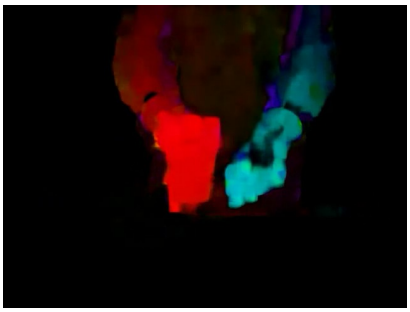
Our network learns to regress the coordinates of the current bounding box to predict the object's position in the next frame, given the optical flow between the two. This section explains our choice of optical flow, the overall algorithmic pipeline, the network architecture, and training regime.

A. Optical Flow

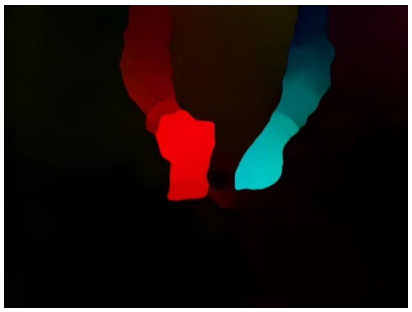
Optical flow between two images is the change in the feature points in one image to the other, in both magnitude and direction. It is a well established way of modelling the motion of an object between frames in a video. Optical flow is usually represented by a two channel image, where one channel represents the magnitude of the movement in the x direction and the other the y.

The decision to choose optical flow as the input to the CNN lies in our belief that when raw images are fed to a neural network for tracking objects, it implicitly learns something similar to the optical flow. Feeding the optical flow explicitly as input helps us make the network simpler and easier to train. Additionally, using a robust optical flow computation method which can handle fast motion well, can help us eliminate a major drawback of existing neural network based object tracking methods.

We experimented with some established optical flow computing algorithms. OpenCV [9] has a built-in function for computing dense optical flow based on Gunner Farneback's algorithm [10]. But the algorithm is inaccurate and fails to handle fast motion. Flownet2 [11] is the second iteration of a deep learning based optical flow algorithm. Though its output is significantly more accurate, the computation time is drastically higher.



(a) Gunner Farneback (OpenCV)



(b) Flownet2

(c) Brox

Fig. 3: A qualitative comparison of the optical flow algorithms on fast-moving hands. The color represents the direction, while the brightness represents the magnitude.

	OpenCV (CPU-only)	Flownet2 (CPU+GPU)	Brox (CPU+GPU)
5min video	∼5mins	∼60mins	∼4mins

TABLE I: Time comparison of the optical flow algorithms.

Brox algorithm [6] is a good middle ground between [10] and [11] in terms of accuracy and ability to handle fast moving objects as shown in Fig. 3, but taking significantly lower time to compute compared to [11] as shown in Table I.

B. Pipeline

We aim to predict the location of the object in the next image, given its location in the current one. This can be formalized as trying to predict B_{t+1}, from I_t, I_{t+1}, and B_t.

The first step in the process is computing the optical flow between frames I_t and I_{t+1}. This optical flow, F_t, which is the same width and height as I_t is cropped to solely the region inside of B_t. This allows the network to focus on only the information relevant for predicting the object's motion. We found that providing a region larger than the bounding box did not yield any performance gains. This crop is then resized to a 128x128 pixels using bilinear interpolation. This is necessary as our CNN contains fully-connected layers and therefore requires a constant-size input. From this optical flow input, the CNN predicts $\Delta pred = (\Delta x_{pred}, \Delta y_{pred}, \Delta width_{pred}, \Delta height_{pred})$, which represents the predicted change from B_t to B_{t+1}.

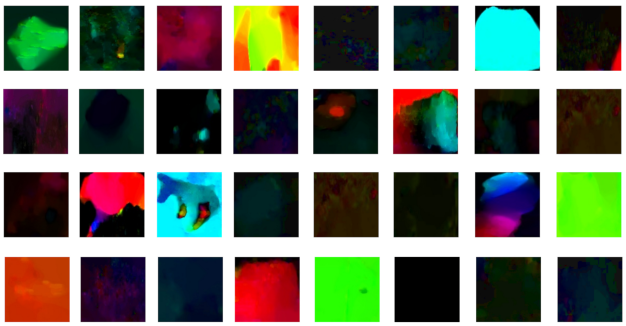


Fig. 4: Examples of the input data to the network, optical flows which are cropped on the groundtruth and resized to 128x128 pixels. They are visualized in the HSV colorspace.

Thus, in the next iteration, the bounding box is updated:

$$B_{t+1} = B_t + \Delta_{pred}$$

C. Network Architecture & Training

A convolutional neural network is used to learn the mapping from optical flows to bounding box motion, $\Delta pred$. We utilize a 12 layer neural network which has 6 convolutional layers and 4 pooling layers in between the convolutions. Two fully connected layers serve as the last layers of the network. After each of the convolutional and fully connected layers, we apply a rectified linear unit (ReLU) non-linearity to the output. We determined these hyperparameter choices while trying to overfit on a small sample of training data and running experiments on the validation dataset.

There has been a great deal of concern in the tracking community about overfitting to the dataset, given how comparatively-few videos the approaches are tested on. Given this, the organizers of the VOT challenge [1] have prohibited the use of tracking datasets OTB, VOT, ALOV, NUSPRO, and TempleColor for training due to concerns about overlapping or similar data. Therefore we train on ImageNetVID, which is a diverse object detection from video dataset that consists of more than 400,000 frames across over 3,700 videos. Each training input consists of the cropped and resized optical flow as shown in Figure 4 and the delta between the current and next boxes. We are training with teacher forcing, which means that errors are not allowed to accumulate during training. Therefore, we simply use the ground truth bounding boxes to crop the optical flows and to compute the $\Delta pred$ $(\Delta x, \Delta y, \Delta width, \Delta height)$ values for training. The delta values are all normalized prior to training, so they have a mean of zero and variance of one. L2 regularization is used on the network weights to prevent overfitting. Like many other regression approaches, we uses SmoothL1 as the loss function between each predicted delta value and the groundtruth. Training is done using the Adam optimizer [12] and we use a learning rate scheduler which

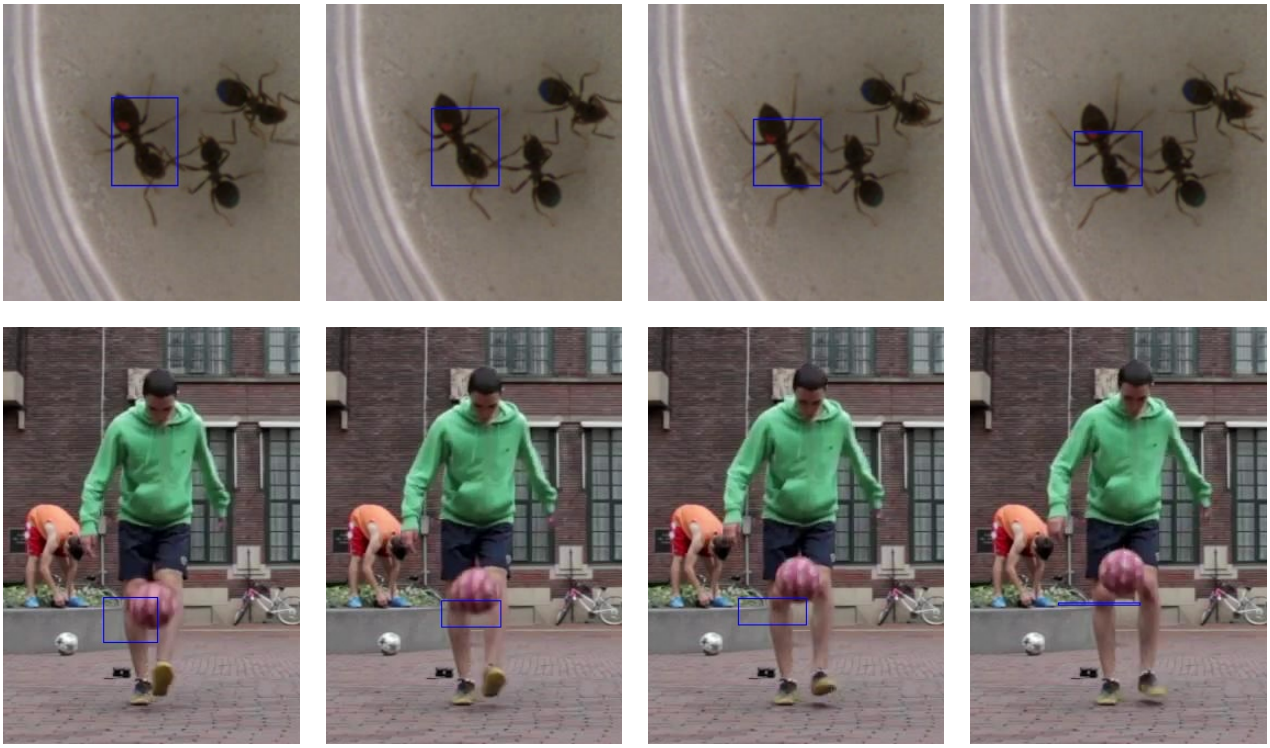


Fig. 5: Results on the VOT Challenge Dataset. The tracker was initialized on the leftmost frame of the sequence.

decreases the learning rate by a factor of ten if the validation accuracy goes down for five consecutive epochs.

IV. Results

We notice that during validation, the network predicts very small values as seen in Fig. 1. This is likely because the distribution of deltas in the training set are predominately small and seem to have a significant peak near zero. This also could be a bias in the way the magnitudes of the optical flows are computed. In any case, it means that the network is struggling to learn even the distributing of the training set.

It is unsurprising, therefore, that the network generalizes poorly to the VOT challenge dataset [1]. When exposed to these sequences which often have far faster motions than what it was trained on, ImageNetVID, it is predicting spuriously large values. In many cases this results in the bounding box decreasing to zero in one dimension. An example of this can be seen in the second series in Fig. 5. Because of this, the network rarely tracks the object for more than a few frames.

V. Conclusion

This paper presents a CNN-based tracker which learns to track objects from optical flow. Our results demonstrate that we likely need to perform additional hyper-parameter tuning to get more predictable results. However, we still believe that optical flow can be a useful input to a tracking algorithm. If it is shown that optical flow alone is not sufficient to achieve competitive tracking results, then providing it as an additional input to an image-based tracker could also be explored.

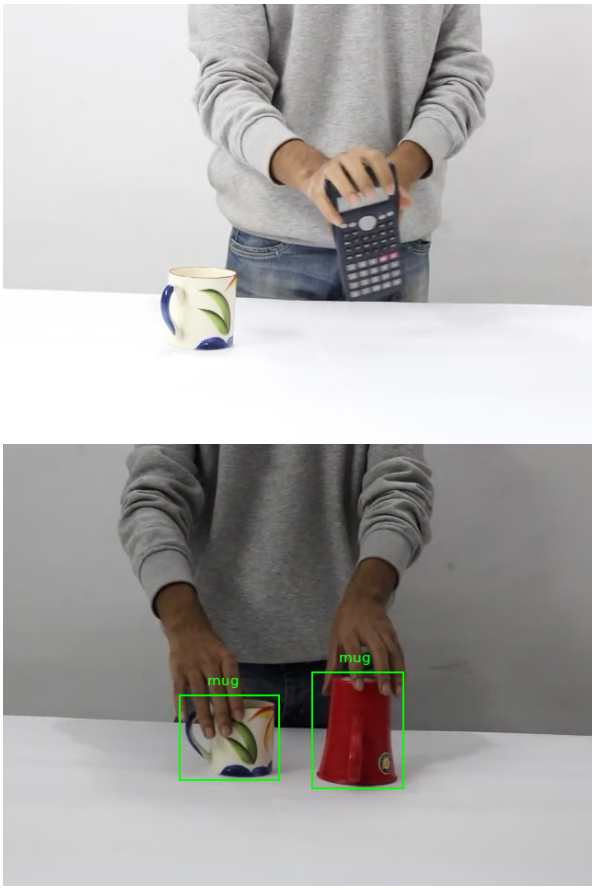


Fig. 6: New dataset with annotated object occlusions

Additionally, through this work, we have identified several areas which we think would be worthwhile to explore.

Currently, our approach only reasons on pairs of frames and does not consider the entire sequence. While this makes it simple and easy to train, it is possible that building longer-term models could make the approach more robust. This network could be combined with a complementary recurrent network to capture the motion model of the object in order to make the predictions more robust. This has been demonstrated to be useful in works such as [13], [14] where multiple network's prediction are combined.

Generating synthetic training data could possibly improve the tracker's robustness in challenging situations. Currently, the training deltas and flows only come from consecutive frames of a video, thereby resulting in deltas which are very small in magnitude. However, it would be straightforward to sample the videos more sparsely in time, to have larger movements between frames. Additionally, we could generate artificial occlusions prior to computing the optical flow. This could be done with simple geometric masks, or with images taken from an instance segmentation dataset.

Finally, object occlusion of any sort is something which trackers generally ignore. Being able to track an object when the human hand holding the object completely occludes the object is important in human-robot interaction. Being able to capture the motion model as stated above can be helpful in interpolating the motion of the object as it gets occluded. We have started looking into this aspect.

Currently no dataset exists having annotated occluded objects in order to train a network in such cases. Artificially occluding objects can help us in it, but a dataset is a useful thing to have. We have recorded about 300 videos which add up to five hours of video data depicting object occlusion from the perspective of a robot when interacting with a human as shown in Fig. 6.

Acknowledgments

We are extremely grateful to Mathis Petrovich for his assistance with this work and Dr. Martial Hebert for his mentoring. David and Vivek would respectively like to thank the National Science Foundation (NSF) and Federation of Indian Chambers of Commerce & Industry (FICCI) for their financial support. Finally, we would like to acknowledge Rachel Burcin, Dr. John M. Dolan and Ziqi Guo for their tireless commitment to the Robotics Institute Summer Scholars program.

References

[1] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, Nov 2016.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV 2016 Workshops*, 2016, pp. 850–865.

[3] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European Conference Computer Vision (ECCV)*, 2016.

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, 2010.

[5] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *International Conference on Computer Vision (ICCV)*, 2017.

[6] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, March 2011.

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2758–2766, 2015.

[8] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 2756–2759.

[9] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[10] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370.

[11] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," 12 2016.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 300–311, 2017.

[14] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Road Marking Quality Assessment System Through Semantic Segmentation

Shaden Shaar[1] and Jahdiel Alvarez[2]

Abstract— **Detecting road markings' quality is an important task to maintain driver safety, their inspection and evaluation is mostly done manually by human inspectors. These markings are regulated by governmental standards which dictate the optimal characteristics of such markings in order to have safer roads. Nowadays, specialized inspectors traverse the roads assessing each standard road markings quality and compliance with regulations. However, human inspection for this task leads to subjectiveness and non-uniform classifications. To automate this process, road marking detection and classification is an essential step. Many previous approaches, use traditional computer vision methods which use hand-made feature detectors in order to utilize the geometry and shape of the markings from the images. We propose a deep learning approach which detects road markings, classifying the quality based on similar metrics to US governmental standards. A fully convolutional network is utilized to perform semantic segmentation on images obtained from cell phones. We classify each pixel into background, road, or road marking. Once we detect the markings we classify them, using the same neural network, leading to novel results in such task. From the authors knowledge this is the first implementation on road marking quality assessment using a computer vision and deep learning approach.**

I. INTRODUCTION

In the United States, the Federal Department of Transportation (DOT) in conjunction with each state's DOT are responsible for ensuring safety and efficiency in transportation systems. Their task consists of inspecting the city's transportation infrastructure, which includes monitoring the state of road markings to assure that it meets the required standards. This monitoring is mostly conducted by human inspectors, which is an expensive and labor-intensive process. In order to automate the road marking inspection procedure, a road marking detection and classification system is required.

Road markings are essential to driver safety, providing guiding lines to ensure proper traffic flow, specifically during the night. Also, given the rise of autonomous vehicles it is important for such systems to be able to detect all road markings and, in our context, classifying the quality of the markings ensures the DOTs can maintain them efficiently, which leads to better performance by the autonomous vehicles, due to the fact they use lane marking detection algorithms.

The detection of road markings is not a new research topic, therefore many of the methods used to detect lane markings are based on traditional computer vision methods and most require knowledge of the sensor models in the vehicle, leading to very specified methodologies. In [1], [4], [10] they take advantage of the camera model by using the camera's angle with respect to a plane parallel to the road, as shown in Fig. 1 to create an Inverse Perspective Mapping (IPM) and run detection after. These approaches do not work on images for which one does not have the camera model, limiting the generalization of the algorithm. In [11] they obtain an IPM by detecting road edges and then using Hough transforms to linearly fit the road boundary and use those fitted lines to obtain the homography matrix. Even though [11] does not use the camera model, it only works with straight roads which are only a subset of all the types of lane markings. Other methods have used spline models and quadratic curve models also. The problem with most of these methods are that they do not perform well in complex environments, and road scenes are some of the most complex scenes particularly in urban environments.

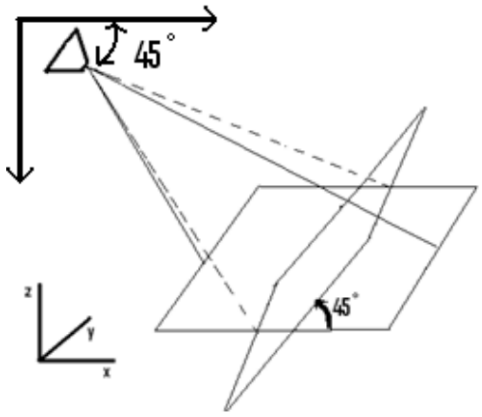


Fig. 1. Traditional vehicle-mounted camera model. Data Source: [9]

Our main approach focuses on the generalization of lane marking detection, which is very difficult through the use of traditional computer vision methods. With the advent of deep learning, and the rise of state-of-the-art architectures for the semantic segmentation task we look into a deep learning approach for the markings detection given recent deep learning approaches have been outperforming traditional computer vision methods, such as in the object detection and semantic segmentation tasks [2], [3], [6]. While traditional approaches work well when the markings are under good, clear lighting conditions, their performance decreases when these conditions change. For instance, changes

This work was supported by Carnegie Mellon University's NAVLAB

[1]S. Shaar is a computer science student in Carnegie Mellon University, Qatar.

[2]J. Alvarez is a Software Engineering student at University of Puerto Rico, Mayaguez.

in the reflectivity or the illumination, a fade in the color, or an occlusion by a vehicle would cause such decrease. Learning based approaches such as machine learning and deep learning methods have become more prominent recently as they are able to learn and extract higher level features from the input in their particular tasks, leading to good performance in difficult scenarios such as poor lighting and occlusions. Our model learns to extract such high-level features, providing detections of road markings.

Recent methods integrate deep learning to their computer vision algorithms such as in [10], obtaining better performance but still limited in terms of generalization given you need knowledge of the sensor models. In [8] a general approach is used through a convolutional neural network (CNN), specifically the object detection framework, Faster R-CNN. In this approach patches of lane markings are detected and using those detections various lines are fitted through them. Given our goal is to detect the actual marking, this approach did not seem appropriate, although it proves the efficiency of learned models in the lane marking detection area.

In order to detect the actual road markings with their contour we took the semantic segmentation approach, where you classify each individual pixel in the image. Modern semantic segmentation architectures have accomplished state-of-the-art results recently [2], [5], [6]. These networks provide great results and generalize really well compared to prior methods. They are able to predict correctly even with poor lighting, rare viewing angles and occlusions, making them good candidates to detect road markings. In order to keep our algorithm as much agnostic as possible, we pass our images through the FCN-8s architecture [6] and learn the weights for road marking segmentation.

We used transfer learning on the FCN-8s model to classify the three desired classes (road markings, road and neither). The model was trained on the Mapillary dataset and we achieved a meanIOU of 70.95% with a class IOU of 40.33%, 77.02% and 95.49% respectively.

The outline of the rest of this paper is as follows. Section II describes the data set that we used, section III describe our approach, section IV reports our experiments and results and section V summarizes our method and emphasizes our future work.

II. Mapillary Dataset

There are many data-sets that targets object recognition on roads such as Mapillary, CityScape, etc. But included in some of those dataset, we can find labels that correspond to lane markings and roads. We used the Mapillary dataset and built upon it to create a modified dataset that fits the premise of our problem. We wanted to create a dataset that can train a model to classify road markings, roads and non markings/roads.

Mapillary dataset is built from images collected from individuals who collaborated in collecting street level images of roads. The images where taken from multiple countries around the globe. In figure 2 we see the heat map representing all the countries that the images where taken from. There are 20,000 images annotated. The annotation process was done manually through a fine-grained style that uses polygons to create the contours and the masks around the targeted areas. The annotation represent 66 different classes including cars, cross-walk, road, people, etc [7]. In figure 3 we can find all the classes that are used in the dataset.

The images try to capture a variety of scenarios, such as snow, sunny, rain, etc. The images also capture multiple and different lane marking schemes of the different countries. Therefore, we notice that lane markings that are captured in images cohere to different governmental standards around the glob. This gives an advantage as we could train a generalized model that can work in multiple locations around the world. The entire dataset consists of 18,000 training images and 2,000 validation images which are annotated and 5,000 testing images that are not annotated [7].

To prune and modify the dataset we selected images that contain a form of lane markings. We merged all the classes that correspond to lane marking-general, lane marking-crosswalk, service lane, crosswalk-plain and bike lane to represent road markings. We also wanted to classify and predict roads. We removed all the annotated images that does not include roads or road markings from our modified dataset. We also had to remove all the image containing snow as the Mapillary dataset colored snow and Lane Marking - General as the same color which can confuse the neural network.

After the pruning with the dataset we ended up with a small confide dataset that represent only the three desired classes. We ended up with 17,742 annotated images that contain all of the three classes and does not have snow in them. We then split the dataset to the same rations that the original Mapillary dataset was in. The split for training, validation and test was 72%, 8% and 20% respectively.

In Table I we can represent, on average, for any given image in the modified dataset the percentage of pixels that belongs to one of the three classes. The image sin the dataset had different sizes as well; therefore, while training we had to re-size the images.

Class	Road markings	Road	Non markings/roads
Percentage	3.2%	18.9%	77.9%

TABLE I. Represents the percentage of pixels representing a specific class on average in any image in the modified dataset

III. Transfer Learning Approach for Lane Mark Detection

Our main approach to this problem is to use transfer learning on well known semantic segmentation models to classify each pixel into the three classes. Famous semantic segmentation models are like FCN, VGG, DeepLab, etc. These models perform pixel wise classification. That is, after performing the convolutions they perform upscale or

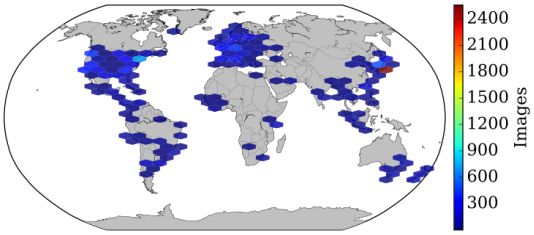


Fig. 2. Countries from which the images in the Mapillary dataset where collected from. Data Source: [7]

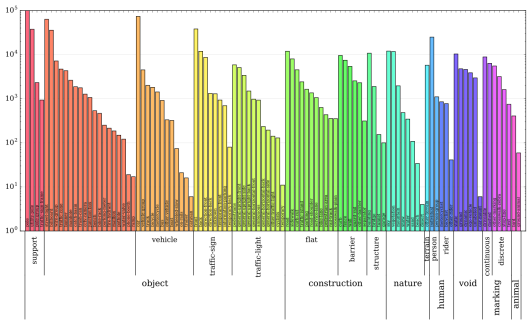


Fig. 3. All the classes that are included in the dataset. Data Source: [7]

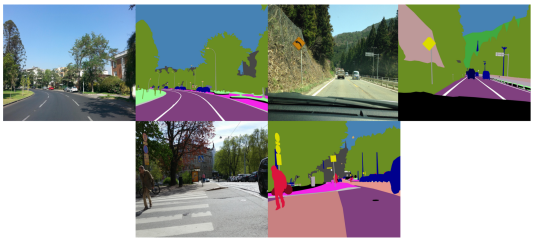


Fig. 4. Examples of images from the original Mapillary dataset. Data Source: [7]

deconvolutions where they create prediction with the same size as the input. the first model to deploy this method is the fully convolutions networks, FCN. [6]

A. Model Description

We used the FCN-8s model to predict the three classes. Figure 5 represent the the FCN-8s model. The FCN-8s consists of 7 2D convolutional layers with pooling in between. The special feature that this model has is that it has 2 skip layers starting from pool-3 and pool-4. These skip layers are used to create the output of the model before the deconvolutional layers. A simple adding function is used to combine pool3, pool4 and conv4. This vector could be considered as an embedding of the image. After we obtain this vector an up-scaling is performed on it to create the logits on each pixel.

Due to the nature of the model, the model is huge as the fully convolutional layer tries to project the embedding into the original image size. This cause the model to have around 134,000,000 parameters all together. We thought about freezing the encoder part but that would not work to our advantage as the upsizing layers does not have non-linear activation functions. They are used as affine transformations only. Therefore, we need to train the model in entirety due to that reason and the presence of the skip layers.

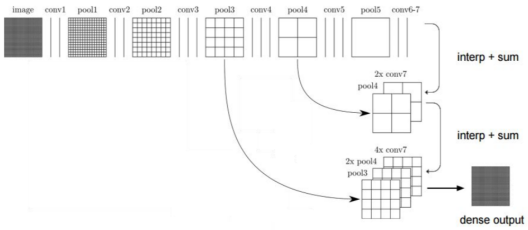


Fig. 5. Fully Convolutional Network FCN-8s

B. Weight Initialization

We used the pre-trained weights of the state-of-art FCN-8s that is trained on the VOC data set. The initial model was trained to detect and identify all different road objects from cellphone images. The VOC dataset is a huge dataset that tries to identify cars and people only on roads. Using these pre-trained weights can shorten the convergence time and give good initialization to out model. Since the scope of the problem is still using the road view images then we assume that the conv layers in the models will be able to capture important features in the images. Therefore, we wanted to use a better initialization of the encoder and then train the entire model end-to-end to predict and identify the three classes.

We also tried to have better initialization through training the network on a small subset of training set. Due to the huge size of the network, if we trained directly on the entire dataset the convergence rate will be very low; as the network will be confused by the different images. We trained the model on 16 images for 300 epochs to get very accurate results on a small dataset. Once the model converged to a small enough loss then we used the entire training set to train the model. This cut out training time by fourth.

IV. Experiments and Results

A. Data Augmentation

We trained the model on downsized, or scaled down sizes of the, images to 500x500 pixels. The reason for that is due to the difference in the image sizes of the dataset we had to make it uniform to perform batching. Also, as mentioned in Table I we see that only 3.2% of any image represents road markings. If we had a 3000x2500 image then it will be harder for the model to accurately predict all the pixels correctly. Hence downsizing the images can give faster convergence rate while training.

None of the images were pre-processed and no data augmentation was done throughout the training and testing process. The training was done on the images as is. A possible way to augment the data is instead of downsizing the images we could use smart cropping. The problem with cropping is that we might end up with crops that do not contain all three classes. Instead we could perform cropping

that targets the bottom of the images as roads normally are in the bottom of the images. This part is definitely considered for future work.

B. Detection Experiments

The training was done on a single NVIDIA GTX TitanX GPU with 12GB of memory. The computational capacities were low therefore fine-tuning an already existing network was the best solution.

We trained for the model for 75 epochs with adadelta optimizer with 0.01 learning rate. We used categorical cross entropy loss and normal accuracy as a metric when we were training the model. We performed a loss of 0.1143 on training set and 0.1499 on validation set.

C. Evaluation on Test Set

As an evaluation metric we used intersection over union, IU, metric. The IU is a standard metric when performing semantic segmentation on images. Most papers measure the accuracy of their model using this metric. The IU score for a class c is calculated by Eq. 1.

$$IU_c = \frac{true\ positive}{true\ positive + false\ positive + false\ negative} \tag{1}$$

As for the overall score for all the classes, meanIU, is calculated by Eq. 2. The weights of the classes are considered in the final calculation of the accuracy of the model. This is important because as shown in Table I we see that road markings occupy a very small percentage of an image. Hence, it is important to weight the classes.

$$meanIU = \sum_{class\ c} weight_c \times IU_c \tag{2}$$

Based on the two metrics described in Eq. 1 and Eq. 2 we calculated the IU of each class and the meanIU of the all the classes combined on the test set. We were able to achieve a meanIU of 70.95%. The individual classes IU is in Table II. You can check the inference on some of the images that were in the test set in 6

Class	Road markings	Road	Non markings/roads
IU	40.33%	77.02%	95.49%

TABLE II. Classes individual IU

From the results of the individual IU in Table II obtained on the test set we see that the model performs relatively well on the road and non marking/road classes; however, not as good on the road markings. The reason behind that is the fact that in any image on average only 3.2% of the image is road markings. Add to that to the downsizing of the images, the downsizing algorithm might be simplifying the road markings too much causing the inaccuracy while training. This makes it hard for the model to detect the lane markings.

From the inference images in Fig. 6 we notice that the model did indeed learn that the lane markings and roads are

Fig. 6. Inference on some images from the test set

located in the bottom of the image. It still need to accurately separate the roads from road markings. According to the III we see that most of the false rejections of the road markings are going to roads. We believe that with bigger computational power we will be able to achieve better results. The loss still has not converged; therefore, with more computational time we could achieve better results.

True/Predicted	Road markings	Road	Non markings/roads
Road markings	$1.6x10^7$	$1.1x10^7$	$0.5x10^7$
Road	$0.6x10^7$	$15.4x10^7$	$2x10^7$
Non markings/roads	$0.1x10^7$	$0.8x10^7$	$73.5x10^7$

TABLE III. Confusion matrix on the test set. Each value represent the number of pixels. Column is prediction and row is true label

V. Conclusion

Using our transfer learning approach we were able to get a meanIU on the test set of 70.9%. This indicates that our model indeed is somewhat able to differentiate between road markings, roads and everything else. However we notice that the IU of the road marking class is about 40.33% which could be improved upon. One method that could be used to improve upon the IU of the classes is through better data augmentation. As suggested in the Section IV subsection A, we could do smarted cropping of images to make sure that the training images has some portion of the three classes. We could also perform cropping and accept cropped images that has at least a threshold percentage of each class.

This entire work was done to perform semantic segmentation to only detect lane markings. We would like to extend our research to perform quality assessment of the lane markings. But due to the lack of presence of a dataset that gives quantified value to road marking quality we would probably use synthetic dataset to train a model that can give the quality of the road markings. Another approach would be to perform stratified active learning on the unlabeled dataset and only label very few instances that could represent a group pf similar lane marking images.

ACKNOWLEDGMENT

We want to thank Dr. Christoph Mertz for his exceptional mentoring and guidance through the RI Summer Scholar 2018 program for helping us throughout our day-to-day research experience. Special thanks, to the RI and RISS personnel for providing a state-of-the-art summer internship. We

also would like to thank our housing universities University of Puerto Rico and Carnegie Mellon University at Qatar for their financial support.

REFERENCES

[1] Mohamed Aly. Real time detection of lane markers in urban streets. *CoRR*, abs/1411.7113, 2014.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] B. S. Khan, M. Hanafi, and S. Mashohor. A real time road marking detection system on large variability road images database. In Rayner Alfred, Hiroyuki Iida, Ag. Asri Ag. Ibrahim, and Yuto Lim, editors, *Computational Science and Technology*, pages 31–41, Singapore, 2018. Springer Singapore.

[5] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579, 2015.

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

[7] G. Neuhold, T. Ollmann, S. R. Bul, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5000–5009, 2017.

[8] Yan Tian, Judith Gelernter, Xun Wang, Weigang Chen, Junxiang Gao, Yujie Zhang, and Xiaolan Li. Lane marking detection via deep convolutional neural network. *Neurocomputing*, 280:46 – 55, 2018. Applications of Neural Modeling in the new era for data and IT.

[9] M Venkatesh and P Vijayakumar. A simple bird's eye view transformation technique. *International Journal of Scientific and Engineering Research*, 3(5), May 2012.

[10] Y. Y. Ye, X. L. Hao, and H. J. Chen. Lane detection method based on lane structural analysis and cnns. *IET Intelligent Transport Systems*, 12(6):513–520, 2018.

[11] Z. Ying and G. Li. Robust lane marking detection using boundary-based inverse perspective mapping. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1921–1925, March 2016.

Hardware Acceleration for Sensor Data Fetch

Junyan Su[1]

***Abstract*— This paper considers a logic hardware design for sensor data fetch. Most control systems have multiple sensors to get the feedback from the environment. To fetch data from all the sensors, a CPU has to consume time waiting. However, on real-time systems like robots, the computational resource is limited and CPU also performs compute-intensive algorithms like sensor fusion. This paper provides a logic-circuit-level design to fetch sensor data with the smallest CPU intervention. Performance is evaluated empirically with PSoC5LP and MPU9250. Results show a significant improvement in sample rate. This leads to more advanced algorithms, like data fusion, to be processed with higher performance. In the future, this design may be integrated as an IP core (semiconductor intellectual property core) and used in microcontrollers that need I^2C block. so that we can fetch sensor data small CPU intervention.**

I. Introduction

Most control system like robots needs sensors to get the feedback from the environment to make the decision. In most cases, the system uses multiple sensors to acquire the information on different aspects of the world and make use of those hybrid data to reduce estimation error and make smarter decisions.

How to fetch data from all those sensors fast is essential in an embedded system in most cases. Many protocols like Inter-Integrated Circuit(I^2C)[1] and Serial Peripheral Interface (SPI) have been used widely for data communication between microcontroller and sensors. In a multi-sensor system, I^2C is preferred since it only requires two general-purpose I/O while SPI requires more pins to communicate with multiple devices.

However, I^2C block in typical microcontroller use software to perform I^2C processing which requires CPU core to manipulate some control registers inside I^2C block. Due to the speed limit of the I^2C protocol, the CPU will waste a lot of time waiting. In a real-time system, CPU resource is quite limited and should be used for more compute-intensive algorithms like data fusion and high-level control algorithms.

To fetch data with smallest CPU intervention, a logic hardware design is proposed to parallelize the I^2C process. We use PSoC5LP and its on-board Universal Digital Block to design the logic circuit with additional error handle functions and user-friendly APIs. Performance is tested with MPU9250 with data fusion algorithms Mahony Filter[2] and Madgwick Filter[3]. As a result, we achieve to reduce the CPU intervention time dramatically from 700 microseconds to only 1 microseconds.

[1]Junyan Su is a student at School of Information Science and Technology, ShanghaiTech University, Shanghai, China. This work is done during Robotics Institute Summer Scholar program in the Robotics Institute, Carnegie Mellon University. sujy@shanghaitech.edu.cn

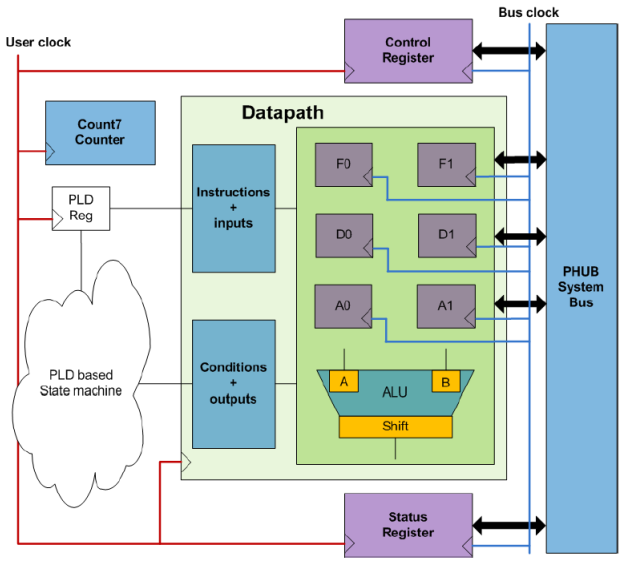


Fig. 1. The main blocks in a UDB from a high level [4]

II. Background

A. Programmable Logic Device

PLD (Programmable Logic Device) is an electronic component used to design digital circuits. Designers can use HDLs (Hardware Description Languages) like Verilog and VHDL to configure a PLD to achieve certain logic functions. Since it is programmable and re-configurable, it is easy for circuit developers to prototype a new design. After testing on prototype, designers can easily translate their source codes into an IP core (semiconductor intellectual property core) which can be used by other IC manufacturers.

We will use one of the variants of PLDs, called UDBs (Universal Digital Blocks) to prototype our design. A UDB is a flexible, programmable digital block inside a PSoC device [4]. The main blocks in a UDB from a high level is shown in Figure 1. Like other variants of PLDs, a developer can use Verilog to configure UDB for a certain logic function. Therefore, after testing our design, it can be easily translate to a IP core and used in future SoC.

B. I^2C protocol

I^2C (Inter-Integrated Circuit) is a widely-used, low-speed serial computer bus for low-speed peripheral ICs to communicate with microcontrollers within a short distance. A typical I^2C design has two bus clock(SCL) and data(SDA) with 7-bit addressing.

There are two kinds of nodes on the bus: master and slave. The master node is in charge of the SCL line and will

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

Fig. 2. Typical protocol to communicate with a sensor via I^2C from [5]. The microcontroller writes a byte to indicate from which address (RA) inside sensors it would like to read/write. Then the microcontroller reads/writes several bytes start at the address (RA) indicate before.

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	MPU-9250 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

Fig. 3. Description for the Figure 2

decide when to start or stop the communication and which slave should respond to the communication. Slave node will detect the SCL line change and respond when corresponding address detected.

As can be seen above in Figure 2, to start the communication, the master generates a START condition on the bus with the slave address as well as a write/read bit following. Then after waiting for the ACK signal from the corresponding slave node, the master node will start to read/write data. Note that if the start condition is NACKed, the master node will consider the slave node not available and generate the STOP condition immediately. For every byte, 1) if the master node sends data to the slave node, the master node will wait for ACK and then start next transmission. 2) if the master node receives the data from the slave node, the master will generate an ACK every time it receives a byte successfully. On the last byte, it will generate a NACK to tell the slave node not to send byte anymore. Finally, to complete the transmission, the master node generates a STOP condition on the bus.

From the top view, to read/write data from/to the typical sensors like MPU9250, the microcontroller (master node) writes a byte to indicate that from which address (RA) inside sensors it would like to read/write. Then the microcontroller reads/writes several bytes start at the address (RA) indicate before.

I^2C has several advantages over other types of computer buses[1]:

- *simple*:I^2C only has two lines for transmitting data for multiple peripherals and the implementation is also simple.
- *well-known and Universally accepted*: most sensors have built-in blocks for communication with microcontrollers.
- *plug& play*: Since there are only two lines, additional sensors can be easily added to the original system with little change.
- *cost-effective*: Cost to build an I^2C block is quite low.

However, the main shortage of I^2C is the limited speed. For example, the I^2C fast mode runs at 100kHz which means no more than 12.5kByte/s communication rate. Any package greater than 12.5 bytes will reduce the sample rate down to less than 1kHz. In fact, the size of the whole sensor data package from one MPU9250 is 22 bytes. In a multi-sensor system, there are many other data coming which further reduces the sample rate. Since it requires the microcontroller to generate different signals, there are a lot of CPU interventions during fetching data via I^2C .

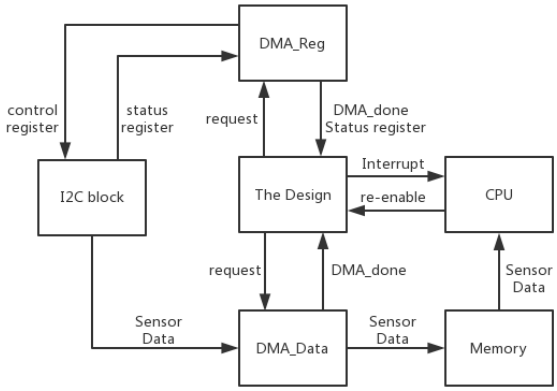


Fig. 4. Design Top View: DMA_Reg is a DMA instance to synchronize the I^2C registers between I^2C and the design. DMA_Data is responsible for transferring sensor data to the memory.

III. The Logic Hardware Design

As stated above, the speed of I2C is quite limited. It limits real-time system like robots to perform advanced algorithms, like data fusion, with high performance. Our design corrects this limitation, saving a lot of CPU time to allow these systems to execute high-performance tasks.

A. Design Top View

As shown in Figure 4, in an I^2C block, there are two types of registers: control register and status register. Control register can be changed by the user (CPU or our design) to send a command to the I^2C block so that it will perform a certain function like generating START or STOP condition on the bus. Status register indicates the state of the I^2C block. The user can check if the transmission is completed or not by reading the status register. Details can be found in [6].

Since the on-board UDBs (Universal Digital Block) have no access to the registers inside the I^2C block, we use DMA (Direct Memory Access) to apply the control register change

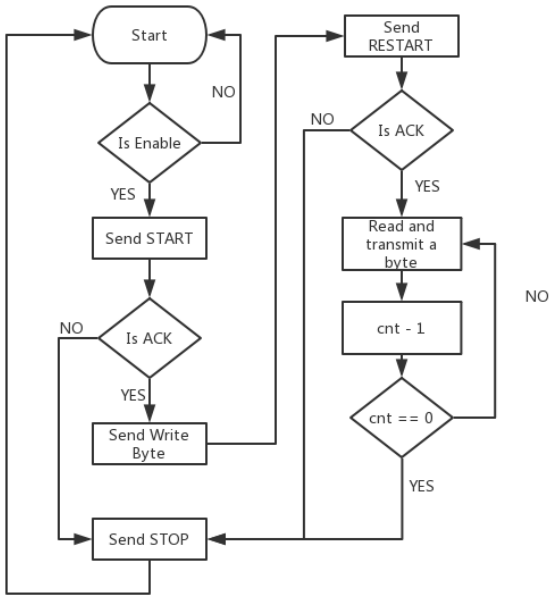


Fig. 5. Flow chart of the design

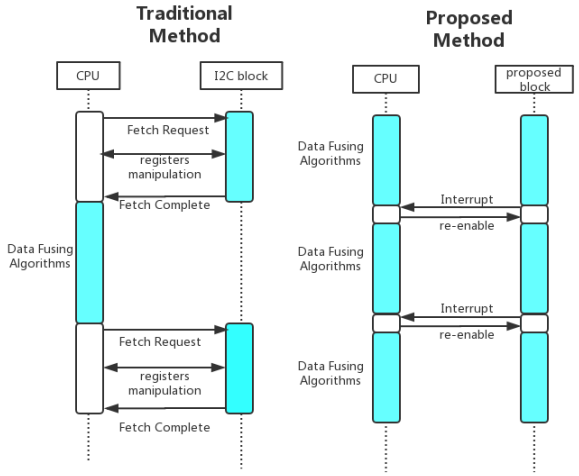


Fig. 6. The sequence diagrams of the traditional method vs the proposed method

to the I^2C block and read the value of the status register of I^2C block. Also, DMA is used to transfer sensor data from I^2C block to the memory so that CPU can easily access it.

In addition, whenever our design finish transferring, it will generate an interrupt and disable itself automatically. Inside the interrupt, CPU may start read from another sensor or simply re-enable to read the current sensor again. This design is useful when the sequence of reading data from sensors matters. We can easily design a FSM (Finite State Machine) to decide the order of how we read data from different sensors.

B. Design Details

Figure 5 shows the logic routine of the hardware design. It is similar to the normal I^2C process routine except the enable signal for the extension to multi-sensor system:

1) Waits for the enable signal from CPU.
2) If enable signal detected, it enters into the transmission status and request the DMA_Reg to change the control register with the value corresponding to the START condition.
3) Read from the status register and waits for the ACK signal.
4) Write with the slave register value to indicate start from which address in the sensor will it reads.
5) Request the DMA_Reg to change the control register with the value corresponding to RESTART condition.
6) Read from the status register to check if it can read a byte from the I^2C block.
7) Request the DMA_Data to transfer the sensor data into the memory.
8) Minus the counter by one and check if it is zero. If not, continue reading.
9) Request the DMA_Reg to change the control register with the value corresponding to STOP condition.
10) Disable the hardware and goes back to the first step.

C. Extend to Multi-Sensor System

Since we introduce an interrupt into our design, it is quite easy for the programmer to define a FSM (Finite State Machine) inside the interrupt. For example, if we want to read from the sensors one by one, when an interrupt occurs, we can simply call a function and set the corresponding parameters of the next sensor and re-enable it.

D. Comparison of traditional and proposed methods

Figure 6 shows the improvement in our design compared to the traditional method. In the traditional method, when needing data, CPU has to manipulate some control registers inside the built-in I^2C block and spend most of the time on it. When interacting with the I^2C block, CPU does no computational work but simply change the value of some registers. However, CPU has to supervise the status register all the time to check if a one-byte transmission is completed. And while CPU doing compute-intensive algorithms, the I^2C block is at idle status and does not perform data fetching task which is a waste of I^2C block resource.

In our new method, the CPU and I^2C block work in parallel. What CPU does is to initialize the block by setting the slave address, data length and data buffer address. After initialization, the block will start data fetching automatically and CPU can do other high-level compute-intensive algorithms like data fusion. Whenever the block transmission completed, it will generate an interrupt and be disabled automatically waiting for CPU to re-enable it. In the interrupt, CPU may change the slave to next slave address and start data fetch from another sensor.

In addition, we provide user-friendly APIs (Application Program Interface) for the programmer to use the design. Additional error handling functions are also provided for further debugging.

IV. Experiment

Performance is evaluated empirically with PSoC5LP and MPU9250. We have implemented two versions: CPU version and hardware version. In the CPU version, we simply subtract the time before calling the function from the time point after calling the function. In the hardware version, we measure the time used in an interrupt. The result shows that we reduce the CPU intervention time dramatically from 700 microseconds to only 1 microseconds.

As far as the multi-sensor system, we test our design on a system with one MPU9250 and four hall sensors and connect that to the same I^2C bus. To fetch data from all sensors, it costs 2000 microseconds in CPU version while only 5 microseconds in the hardware version.

Besides, due to the parallelism, the sample rate also increases twice as many as before which leads to a higher accuracy of data fusion algorithm like Mahony Filter we implement on board.

V. Conclusion

In this paper, we provide a hardware design to offload CPU for the I^2C process. Results show that there is a dramatic reduction in CPU intervention time. Since we use UDB with Verilog to design the logic hardware, this design has the potential to be integrated into any general I^2C block as a new IP core in a microcontroller or SoC(System on a Chip) so that sensor data fetch, especially in a multi-sensor system, does not affect a system to be real-time.

Acknowledgment

This work was supported by the ShanghaiTech University and by the Robotics Institute through the RISS program. The author would like to thank Professor Howie Choset and Mr. Lu Li for their guidance though the program. Special Thanks to Dr. John Dolan, Ms. Rachel Burcin and the RISS team for the support.

References

[1] Philips Semiconductors, "AN10216-01 IC manual," p. 51, 2003.

[2] J.-M. P. Robert Mahony, Tarek Hamel, "Nonlinear complementary filters on the special orthogonal group (jun. 08)," *IEEE Transactions on Automatic Control*, p. 12031218, 2008.

[3] S. O H Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," 06 2010.

[4] C. Court and S. Jose, "Universal Digital Block (UDB) Editor Guide," no. 001, pp. 1–52, 2014.

[5] InvenSense, "MPU9250 Product Specification Revision 1.1," *Product Specification*, vol. 1, no. 408, pp. 1–4, 2016.

[6] C. Semiconductor and S. Jose, "PSoC 5LP Registers," no. 001, 2016.

Functional Trajectory Forecasting and Consistency Guarantees for Self-Driving Cars in Social Settings

Scott Sussex[1], Chiyu Dong[2] and John M. Dolan [3]

***Abstract*— Self-driving cars will be required to navigate urban scenarios such as intersections, ramp merges, and lane changes. These are all highly social environments that require accurate estimates of the intentions of other cars. Increasingly, this problem is being tackled by learning a behavior model from data. We present a new learning-based method which is based on non-parametric regression in Reproducing Kernel Hilbert Space. Our method provides three important contributions:**

- **The use of a prediction function class which can account for the interactions between all relevant vehicles, both autonomous and human-driven.**
- **A novel prediction function where the output is a smooth trajectory over time, rather than a series of discrete points.**
- **A bound on how our estimated trajectory will vary as a function of the time at which it is calculated, providing a measure of forecast consistency over time.**

We use the lane change setting to demonstrate and evaluate our method. Our method outperforms alternative approaches. The results also compare the use of different regularization functions and kernels when applying our method.

I. INTRODUCTION

Intention estimation for autonomous driving is a difficult problem. Intention estimation means forecasting what other cars will do in social situations, for example, intersections, ramp merging, and lane changing scenarios. We focus on the lane change scenario for our experimental work, but our theory applies to all of these scenarios.

Social settings are complex and forecasting requires consideration of the behavior of multiple vehicles, which may interact with the vehicle of interest. The difficulty of estimating vehicle intentions in social settings motivates learning a model of vehicle intention from data.

Learning algorithms output a fixed number of points, but we are interested in forecasting the entire trajectory of a target vehicle, which contains infinitely many points. Existing approaches can use a separate function to smooth across the points they output, in order to find a smooth trajectory for the vehicle. Doing this with a method such as a smoothing spline leads to bias at the endpoints of the estimated trajectory. We present a method that accounts for smoothing within the training process, in order to learn a model that accurately forecasts smooth trajectories for target vehicles. We introduce a two-step RKHS method. The first converts inputs to a parameterization of a trajectory, and the second acts as a smoother which generates a trajectory which is a function of time.

Alternative deep learning approaches will be sensitive to hyper-parameter selection. Methods like grid search or random assignment have high training time and do not give optimal hyper-parameters. We demonstrate how a validation set can be used to analytically optimize one of the hyper-parameters in our method.

Sensor error and change in vehicle positions over time may lead to our model receiving noisy or changing input over time. In a self-driving vehicle, it is desirable to have forecasts that do not erratically change over time, so that the vehicle has a consistent view of the world for motion planning. Existing approaches do not provide any way of quantifying how consistent a forecast will be over time. We derive a bound for how much the estimated trajectory will change with respect to the time at which the forecast is made: a mathematical guarantee of the consistency of our forecaster.

We present two different approaches to regularizing models that use RKHS methods to forecast a complete trajectory, and empirically evaluate them on the lane change scenario. Finally, our empirical results also demonstrate the feasibility of using an RKHS method with functional output in the self-driving setting. We find that our method performs favorably compared to alternative approaches.

II. PRIOR WORK

Some previous works on navigating social driving scenarios do not make predictions about the intentions of surrounding cars. An example is Baker and Dolan's slot-based approach [1], used in the CMU Boss merge planner to select where to change lane in a highway lane-change scenario. Information on surrounding cars' motion is used, as well the distance of the host car from the slot it may merge into. The most likely target slot is selected based on feasibility of the maneuver and the context of the scenario. The approach does not make forecasts about the expected behavior of other vehicles when planning.

Nilsson et al. [2] do not make forecasts about the behavior of other vehicles, but instead formulate lane changing as an optimization problem using Model Predictive Control. The approach is limited since it involves manual tuning of constants and optimizes a hand-designed optimization function, rather than designing a trajectory based on forecasts about the behavior of other vehicles.

A significant number of probabilistic approaches to the social behavior problem have been proposed. Yao et al. [3] search for the k-nearest-neighbors in a lane-change database

[1]Scott Sussex is with The Paulson School Of Engineering And Applied Sciences, Harvard University, 29 Oxford St, Cambridge, MA 02138 USA

[2]Chiyu Dong is with The Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA

[3]John M. Dolan is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA

and use these to generate an estimated trajectory by interpolating. Such an approach may have large bias in sparse areas of the feature space, and does not incorporate smoothing when learning the model.

Several approaches use binary or ternary predictions. For example, Galceran and Cunningham et al. [4] [5] use Bayesian changepoint detection on a car's recent history to forecast whether a car will turn left or right. Dong et al. [6] use a PGM to forecast whether a car will yield or not yield in a lane merging scenario. Both of these approaches could be used as a higher-level module, preceding use of a model that outputs an estimated trajectory.

A number of approaches employ a Partially Observable Markov Decision Process (POMDP). Ulbrich et al. [7] and Wei et al. [8] use real-time belief space search to solve an online POMDP. To achieve real-time performance they needed to use a discrete state and action space. Bai et al. [9] proposed a continuous-state POMDP using a belief tree. It was applied to navigating intersections, but only used discrete actions represented by a generalized policy graph. Seiler et al. [10] proposed an approximate online solver for a continuous-action POMDP, but it was only tested in toy problems. All of these POMDP models need their transition models and reward functions to be designed by hand. Sadigh et al. [11] and Hadfield et al. [12] try to learn these from data using inverse reinforcement learning. Sadigh et al. [13] develop upon this and model interactions with human drivers as an underactuated dynamical system in order to learn autonomous behaviors in social settings. The method is interesting in that it allows the autonomous vehicle to plan for how its own behavior will affect a human driver's future behavior. The method lacks hard safety constraints and therefore performs some dangerous maneuvers during human experiments on a simulator. The method is also unable to handle interactions with more than one human driver, due to both computational and modeling difficulties.

Dong et al. [14] also make predictions about other vehicles using non-parametric regression in RKHS. Their model similarly uses the recent motion of all surrounding cars, but only predicts the start and end point of a lane change maneuver, rather than a full trajectory. They also do not show how to analytically determine hyper-parameters or provide consistency bounds on their forecasts.

Existing approaches to the social behavior problem that attempt to estimate other vehicles' trajectories either do not consider interactions with many vehicles, or do not output estimates that represent the full trajectory as a function of time. Outputting forecasts as a full trajectory is more practical for use in motion planning, since it allows for evaluation of another vehicle's expected position at any point in time. To achieve this with methods that output just a sequence of discrete points would require smoothing. Using commonly used smoothing methods such as smoothing splines will lead to bias at the end-points of the trajectory. Not considering smoothing when learning a model means there is no way to account for the bias induced by smoothing. Finally, no existing methods consider bounding how much the estimate can change over time to provide guarantees regarding the planner having a consistent understanding of the world.

III. METHOD

A. Trajectories in RKHS

A RKHS (**H**) contains a family of smooth functions associated with a particular kernel K. $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$. A function f in **H** can be represented by a linear combination of kernel outputs evaluated at each training set element: $f(\cdot) = \sum \alpha_i K_{x_i}(\cdot)$,

Given a set of labels b and a set of inputs x in a training set, traditionally RKHS methods result in solving

$$\hat{f} = \underset{f \in H}{\arg\min} \sum_{i=1}^{N} (b_i - f(X_i))^2 + \lambda J(f) \tag{1}$$

Where $J = ||f||_H^2$. In this case, the representer theorem holds, which states that the loss minimizing f is of the form $\sum \alpha_i K_{x_i}(\cdot)$. In this case, α can easily be found to obtain the optimum f for the loss function. RKHS methods are nonparametric, and used when we do not know the form of function f but are interested in approximating it given data and with potentially unlimited degrees of freedom. For a further discussion of RKHS methods in relation to vehicle forecasting, see [15].

B. Learning a Smooth Trajectory

TABLE I: Table of Notation

x	$\triangleq$	Training data features of n trajectories.
x'	$\triangleq$	The features of a single input.
b	$\triangleq$	The ground truth label for training trajectories. Contains d points at times along each of the n trajectories. Time points are represented by vector t and are consistent across trajectories.
K	$\triangleq$	Kernel function between feature inputs. When no inputs given, assume the matrix $K(x, x)$
k	$\triangleq$	Kernel function between two times. When no inputs given, assume the matrix $k(t, t)$.
η	$\triangleq$	The model's estimated trajectories. This is a function of time. η_i is the function corresponding to the ith training input.
D	$\triangleq$	A derivative operator for k. Can be chosen to represent arbitrarily many derivatives rather than just the first. Assume k is infinitely differentiable so that D is always a linear operator.
α	$\triangleq$	A matrix of the learned weights in the regression.
$.$	$\triangleq$	Inner product operator.
λ	$\triangleq$	A constant to weight between minimizing error and complexity.

Let $f(x') = \alpha.K(x', x)$. Let our output trajectory be η, where $\eta(t) = k(t,).f(x')$.

When training a model, we are interested in a low MSE (mean squared error) between our trajectories at various time points t_i and the ground truth, where $0 \leq t_i \leq 1$ and $0 \leq i \leq d$. We are also interested in generating trajectories that are smooth, so we use smoothness as a regularizer. We can use the loss function:

$$\frac{1}{nd}\sum_{i,j}(b_i(t_j)-\eta_i(t_j))^2+\lambda\sum_i||D\eta_i||_E^2$$

To minimize our loss function on a training set we solve to find $\alpha = k^{-1}(k+dn\lambda D)kb^TK^{-1}$ (Derivation 1 in appendix). Note that for this loss function, the representer theorem does not hold. Therefore the α found is only the optimal linear solution. However, the regularizer used has clear physical meaning. It seeks functions that have a smooth ith derivative of the trajectories forecasted on the training set.

A more simplistic application of traditional RKHS methods might lead one to use $||f||_H^2$ as the regularization term. For the resulting loss function, the representer theorem does hold, so the α found represents a true optimum. α can be solved for in the standard way, but requires the solution of a Sylvester equation [16] as a result of the smoothing step. The intuitive meaning of the regularizer is also less clear. It minimizes the complexity of f but does not consider the smoothing step in the regularization.

C. Analytical Hyper-Parameter Optimization

λ, the smoothing basis function (k), and the non-parametric regression kernel function (K) are all hyper-parameters. For the first loss function given above, λ can be optimized on a validation set as the solution to the linear equation

$$\sum_j\sum_i dnD_jkb^TK^{-1}K_i^*((b_{ij}^*-k_jkb^TK^{-1}K_i^*) \\ -dn\lambda D_jkb^TK^{-1}K_i^*)=0 \quad (2)$$

where K^* is the kernel matrix computed by inputting the validation set trajectories against the training set trajectories. This can be shown by substituting in $\alpha = k^{-1}(k+dn\lambda D)kb^TK^{-1}$ and differentiating with respect to lambda.

D. Provable Bounds on Change in Predicted Output Over Time

Input to the model will be changing over time. Therefore the estimated trajectory will be changing with time. Let $g(T,t)=\eta_T(t-T)$ be the point estimate gathered by taking in input at time T into the future and then evaluating the resulting trajectory at time t after the estimated trajectory's start point. We aim to bound $g(T,t)$ given $g(0,t)$. In other words, we are interested in bounding a future forecast regarding where the car will be in terms of our current forecast. We are particularly interested in bounding $g(t,t)$ since if we have an unbiased model, this will be the exact position of the target car at time t in the future.

In order to compute a bound we assume Lipschitz continuity of the input with respect to time, i.e.

$$||X_1,X_2||\leq C(T_2-T_1)$$

We argue that this is a reasonable assumption because we know the car has bounded velocity and our sensors likely have bounded error. This would bound how much the input can change with respect to time. C can be determined from data or domain knowledge.

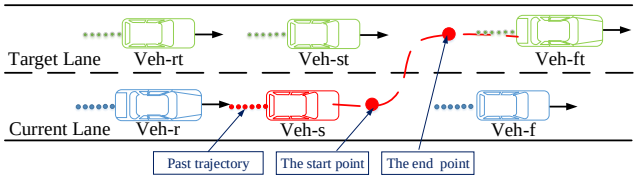


Fig. 1: This diagram from Dong et al. [15] demonstrates the highway lane change scenario and the difference between our two methods. Dong et al. use all the shown cars' past trajectories to forecast the start and end point of the red car's lane change. In contrast, we forecast a full trajectory as a function of time: the dashed red line.

We use a kernel function for smoothing within a trajectory $k(t_1,t_2)$ that is always greater than 0. We smooth at specific time points t_i for $1\leq i\leq d$. For our kernel function that compares input to training set elements, we assume any inverse multiquadratic kernel of the form $K(X,Z)=\frac{1}{\sqrt{||X-Z||^2+r}}$ where z is a training set input, r is a constant, and the squared norm is any norm that satisfies the triangle inequality.

For any regularizer we have presented, given input to the model at time 0 of X_0, we can show that $g(T,t)$ is bounded in the interval

$$\left[\sum_i k(t_i,t+T)\sum_j \min(A_{i,j},B_{i,j}) \quad , \right. \\ \left. \sum_i k(t_i,t+T)\sum_j \max(A_{i,j},B_{i,j})\right]$$

where each Z_j is one of the n training set inputs. and $A_{i,j}=\frac{\alpha_{i,j}}{\sqrt{(||X_0-Z_j||+CT)^2+r}}, B_{i,j}=\frac{\alpha_{i,j}}{\sqrt{(max(0,||X_0-Z_j||-CT)^2+r}}$ (Derivation 2 in appendix) where each Z_j is one of the n training set inputs.

The width of the bound monotonically increases with C as expected. As our bound on input gets wider, our bound on the output also becomes wider. The width of the bound is also monotonically increasing with respect to T as expected. Our bound becomes wider as we look further into the future.

IV. EXPERIMENTAL DESIGN

We carry out thwo experiments to verify our method:

- Report Mean Squared Error (MSE) of our model at different time points along a trajectory forecast, for different regularization options.
- Compare the mean and standard deviation of errors from our method and alternative approaches.

We evaluate the model specifically on the highway lane change scenario. This scenario is shown in Fig. 1. We call the vehicle changing lanes the host car. The input to our model will be the previous trajectory of the host car and all surrounding cars. The surrounding cars include the leading and following car in the current lane, in addition to

TABLE II: Regularizers Used in Experiments

Trad	$\triangleq$	$Tr(\alpha K \alpha^T)$
Di	$\triangleq$	$Tr((\alpha K)^T D_i k(\alpha K))$ where D_i is the ith derivative operator.

the neighboring car in the target lane, and its leading and following car. Our model's goal is to output an estimate of the trajectory the host car will follow as a function of time.

We use a dataset from NGSIM [17], with traffic data from the I80 and US101 highways. The I80 dataset contains traffic data for three 15-minute periods: 4:00 p.m. to 4:15 p.m., 5:00 p.m. to 5:15 p.m., and 5:15 to 5:30 p.m. The US101 dataset also contains three 15-minute periods: 7:50 a.m. to 8:05 a.m., 8:05 a.m. to 8:20 a.m., and 8:20 a.m. to 8:35 a.m. In both datasets, we have the location of each vehicle at 100-millisecond time intervals. We extract lane change scenarios from the dataset in the same way as Dong et al. [15]. This method gives the trajectory of the host car and all surrounding cars for each lane change scenario. The location of every car is recorded between 10 seconds before and 10 seconds after crossing the lane marking.

We measure the error of our model from ground truth by evaluating our estimated trajectory at a set number of time points in the future, and comparing to ground truth at that time. Comparing separately at different points along the trajectory allows us to illustrate how accurate the model is in the near and far future. We compare to ground truth at 1-second intervals from the start of our estimated trajectory. We extracted 543 lane change scenarios. 450 are selected as training data. The remaining 93 trajectories form the test set.

The kernel used to determine K is the inverse multiquadratic kernel with the Frobenius norm, which has been shown to be effective in previous RKHS approaches to self-driving [15]. Forecasts are made using up to the previous 3 seconds of historical vehicle data at 0.1s intervals as the features.

V. EXPERIMENTAL RESULTS

Trajectory forecasts are evaluated using the method with 4 different regularization terms (Table II). For the second term we use i from 0 to 2.

For each regularizer, K is an inverse multiquadratic kernel where the norm used is the square root of the Frobenius norm.

$$K(X_1, X_2) = \frac{1}{\sqrt{||X_1 - X_2||_F + c}}$$

where

$$||A||_F = \sqrt{tr(A^T \cdot A)}$$

For smoothing, k is selected to be a radial basis function.

$$k(t_1, t_2) = e^{-\epsilon |t_1 - t_2|^2}$$

For all regularization terms, we select hyperparameters $c = 0.5$ and $\epsilon = 2$. c was selected based on what was optimized in [15]. ϵ was selected arbitrarily before training, and could have been set to any value depending on the required smoothness of the model's output.

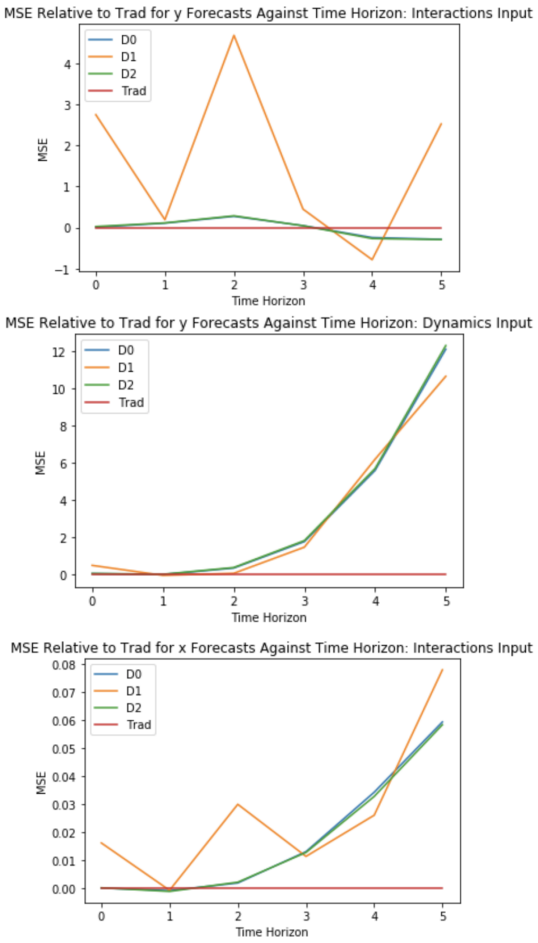


Fig. 2: Graphs of MSE for forecasts in the y and x direction against the point at which the forecasted trajectory is evaluated. All errors are relative to the error of the Trad regularizer. The lines for D0 and D2 are almost indistinguishable because their MSE values are so similar.

We test two different kinds of input X in our experiments. The first is "Interaction", which contains scaled positions of all relevant vehicles at 0.1-second increments over the past 3 seconds, relative to the host vehicle's position at the time of prediction. The second is "Dynamics", which contains scaled positions of just the host car at 0.1-second intervals over the last two seconds. "Interaction" is designed to capture interactions between vehicles, whereas "Dynamics" is designed to mimic a naive method of just considering the host vehicle's velocity at the time of forecasting. All labels (true training set trajectories) were transformed by subtracting the mean training set trajectory. This was done so that regularization would shrink trajectories towards the mean trajectory, rather than towards a stationary trajectory at 0.

We compare the impact of different regularizers and inputs on the forecasting error in directions parallel (y) and perpendicular (x) to the lane markings. Figure 2 shows these results. For Trad, λ is selected by grid search on a validation set, but for D_i, λ is selected by analytical optimization on

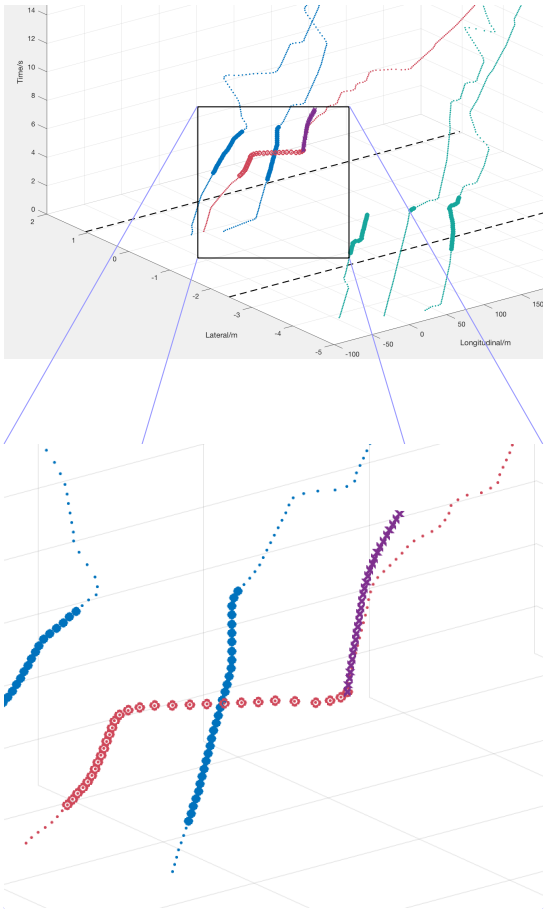


Fig. 3: Visualization of a trajectory forecast on the test set. Red shows the coordinates of the host car. Blue shows the other cars in the current lane. Green is cars in the target lane. Circles (O) are data-points input to the model, while dots (.) are the rest of the trajectory. The dashed black lines are the road markings. The purple points are the forecast.

the validation set as presented above.

We find that in general the Trad regularizer has lowest MSE. In the x direction and in the y direction with an Interaction input, the alternative regularizers perform closely to Trad, while in the y direction with the Dynamics input the Trad regularizer performs far better. This suggests two things: 1) in general, incorporating smoothing information into the regularizer decreases performance; and 2) the performance of different regularizers is dependent on the kernel used. $D0$ and $D2$ perform very similarly, likely because the second derivative of the radial basis function has the same sign as the undifferentiated radial basis function.

A random trajectory is selected from the test set and visualized to qualitatively assess the feasibility of a generated trajectory. The forecast is made with regularizer Traj, using input B in the y direction and input A in the x direction. We make a forecast 3 seconds into the future.

Besides the comparison between different orders of derivative, the proposed method is also compared with two state-of-the-art trajectory estimation methods.

A) The Conditional Neural Process framework (CNP) [18] estimates the stochastic process of the future trajectory. It collects three moments of historical trajectories of all vehicles involved, and predicts the future trajectory. As suggested by the original CNP structure, two three-level fully connected layers are used for observation and generation.

B) An End-to-End LSTM-based approach [19] which directly predicts trajectory of a target, considering surrounding vehicles by social tensor [20]. The LSTM-based approach keeps all historical data of all relevant vehicles and predicts the location at the next timestamp. To make the comparison equivalent, the output LSTM cell is evaluated ten times to predict locations at ten future timestamps. A 1 layer LSTM is used.

In addition, a KNN implementation for the trajectory estimation [3] serves as a baseline method, and the same kernel is used as that used by the RKHS approach in each direction. For the RKHS, we use the Trad regularizer in both directions. We use the Dynamics input in the y direction and the Interaction input in the x direction.

Methods	0s	1s	2s	3s	4s	5s
KNN	0.0000	-0.0020	-0.0686	-0.2773	-0.4981	-0.5374
LSTM	0.2855	-0.0264	-0.3296	-0.5884	-0.7763	-1.2092
CNP	-0.0009	0.0845	0.1808	0.3820	0.3791	0.4439
RKHS	-0.0002	0.0125	-0.0011	-0.1304	-0.2510	-0.2515

TABLE III: Mean errors along the lateral (x) direction at each second after forecasting (0-5) comparing with ground-truth, for each method.

Methods	0s	1s	2s	3s	4s	5s
KNN	0.0000	0.3088	0.7295	1.4579	2.1624	2.5313
LSTM	0.9431	0.7764	0.8807	0.9191	1.0202	1.1601
CNP	0.0051	0.4764	1.2679	1.9950	2.3249	2.4724
RKHS	0.0012	0.2876	0.6260	1.2407	1.8513	2.1388

TABLE IV: Standard deviation of errors against ground truth along the lateral (x) direction at each second after forecasting (0-5), for each method

Methods	0s	1s	2s	3s	4s	5s
KNN	0.0000	0.0250	0.0003	0.0769	-0.1308	-0.4035
LSTM	-1.4577	-2.1003	-4.4286	-6.4974	-9.8511	-17.9320
CNP	-0.0030	-0.2291	-0.5650	-0.7824	-0.9790	1.1256
RKHS	-0.0001	0.0033	-0.0289	0.0637	-0.1279	-0.4262

TABLE V: Mean errors along the longitudinal (y) direction at each second after forecasting (0-5), for each method.

Methods	0s	1s	2s	3s	4s	5s
KNN	0.0000	0.9624	2.5461	4.3528	6.6001	8.9433
LSTM	5.3553	6.3561	8.2254	10.3025	14.6079	20.5533
CNP	0.0096	2.1842	4.8458	7.6852	10.7635	13.9952
RKHS	0.0042	0.8618	2.3333	3.9659	6.0391	8.2210

TABLE VI: Standard deviation of errors against ground truth along the longitudinal (y) direction at each second after forecasting (0-5), for each method.

VI. DISCUSSION

Comparing different kernels, we find that the kernel using the dynamics input works best in the y direction, while also outperforming baseline approaches. We suspect that this is because the car's current velocity is likely the strongest predictor of forward motion. In the x direction, we know this is likely not the case, especially if the vehicle has yet to begin the lane-change maneuver. In the x direction, the kernel considering interactions between vehicles outperforms baseline methods.

In the comparison with previous approaches, as shown in Table III, the RKHS method performs very competitively. In the lateral direction, the LSTM has lower variance but higher bias. In the longitudinal direction, the LSTM has both high bias and high variance compared to the RKHS. RKHS tends to have uniformly lower variance compared to the KNN and CNP, and lower bias. Only in the lateral direction does the KNN have comparable sample bias to the RKHS method. The RKHS method has some bias and variance in errors at the forecasting point, which may appear unusual. This is because of the smoothing taking place. Recall that the RKHS approach is outputting a full trajectory, whereas the alternative approaches are outputting a series of point estimates.

The visualizations demonstrate that the method can generate feasible trajectories. Whilst we were able to compute a consistency bound for a given forecast, we did not include it on the visualization since the bound becomes very large a short amount of time after the time of forecasting. We believe this is because the bound we have derived is very conservative: we consider each term in the sum over training elements separately and form an upper and lower bound on each term.

VII. CONCLUSION

In this work we lay the foundations for using Reproducing Kernel Hilbert Space methods in a setting where we desire a functional output. We focus on the setting of forecasting other vehicles' trajectories for an autonomous vehicle. We show that modified RKHS methods can be used to forecast trajectories with less error than that of alternative methods. We consider different regularization terms and kernels. We derive a bound on the consistency of future forecasts with our current forecast, a first in this setting. However, this bound is very conservative and we hope future work will develop more practical bounds.

While we make progress in applying RKHS to the functional output setting, our loss function still considers MSE at discrete points along the forecasted trajectories. Theoretical work might formulate loss as an integral of error across the entire forecasted trajectory. Finally, we found that having a manually selected kernel function can be restrictive in such a complex setting. We would be interested in seeing the application of deep learning methods to learn richer kernel functions, while maintaining the consistency bound and convex optimization properties of the RKHS approach.

APPENDIX

Derivation 1: show that

$$\arg\min_{\alpha} \frac{1}{nd}\sum_{i,j}(b_i(t_j)-\eta_i(t_j))^2+\lambda\sum_i ||D\eta_i||_E^2 = k^{-1}(k+dn\lambda D)kb^TK^{-1} \quad (3)$$

We can rewrite our loss function as

$$\frac{1}{dn}Tr(||b-(\alpha K)^Tk||^2)+\lambda Tr((\alpha K)^TDk(\alpha K))$$

Differentiate the loss function with respect to α and set equal to 0.

$$-\frac{1}{dn}2k(-k\alpha K+b^T)K+2\lambda Dk\alpha K^2=0$$

The following two identities are used in the above. $\frac{d}{dX}Tr((AXB+C)(AXB+C)^T)=2A^T(AXB+C)B^T$ for the first term. $\frac{d}{dX}Tr(B^TX^TCXB)=C^TXBB^T+CXBB^T$ for the second term.

Rearrange and we get $kb^TK-k^2\alpha K^2=dn\lambda Dk\alpha K^2$

So $\alpha=k^{-1}(k+dn\lambda D)kb^TK^{-1}$. We have a minimum because the loss function is convex in α.

Derivation 2: Assume Lipschitz continuity on the input, with Lipschitz constant C. Assume kernel K is inverse multiquadratic (IMQ). We show that for any regularizer we have presented, given input to the model at time 0 of X_0, we can show that $g(T,t)$ is bounded in the interval

$$\left[\sum_i k(t_i,t+T)\sum_j \min(A_{i,j},B_{i,j}) \quad , \quad \sum_i k(t_i,t+T)\sum_j \max(A_{i,j},B_{i,j})\right]$$

where each Z_j is one of the n training set inputs and $A_{i,j}=\frac{\alpha_{i,j}}{\sqrt{(||X_0-Z_j||+CT)^2+r}}, B_{i,j}=\frac{\alpha_{i,j}}{\sqrt{max(0,||X_0-Z_j||-CT)^2+r}}$

By triangle inequality we know

$$||Z_j-X_1||\le||X_0-X_1||+||X_0-Z_j||$$

for any training set input Z_j.

$g(T,t)=\sum_i k(t_i,t+T)\sum_j(\frac{\alpha_{i,j}}{\sqrt{||X_1-Z_j||^2+r}})$ but by the above, $||Z_j-X_1||\le(||X_0-X_1||+||X_0-Z_j||)\le(||X_0-Z_j||+CT)$ by the Lipschitz assumption. By the reverse triangle inequality we also get $||Z_j-X_1||^2\ge(max(0,||X_0-X_1||-CT))^2$. Using this, we can bound $\frac{\alpha_{i,j}}{\sqrt{(||X_1-Z_j||)^2+r}}$ between $A_{i,j}$ and $B_{i,j}$. The sign of α determines which is the top and bottom of the interval.

This allows us to bound each element in the sum based on the sign of $\alpha_{i,j}$. This gives the interval for $g(T,t)$ above.

ACKNOWLEDGMENTS

The authors would like to thank Rachel Burcin and Ziqi Guo for their work on the RISS program, which is the excellent summer research program at CMU where this work was completed. Thank you to Jordan Ford for discussions during the early stages of the project. Thank you to The Robotics Institute for funding this project.

REFERENCES

[1] C. R. Baker and J. M. Dolan, "Traffic interaction in the urban challenge: Putting boss on its best behavior," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1752–1758.

[2] J. Nilsson and J. Sjöberg, "Strategic decision making for automated driving on two-lane, one way roads using model predictive control," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1253–1258.
[3] W. Yao, H. Zhao, P. Bonnifait, and H. Zha, "Lane change trajectory prediction by using recorded human driving data," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 430–436.
[4] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction." in *Robotics: Science and Systems*, 2015.
[5] E. Galceran, A. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, 2017, in Press.
[6] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *2017 IEEE 28th Intelligent Vehicles Symposium (IV'17)*, Jun. 2017.
[7] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, Oct 2013, pp. 2063–2067.
[8] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based mdp for robust single-lane autonomous driving behavior under uncertainties," in *Robotics and Automation (ICRA) 2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2586–2592.
[9] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.
[10] K. M. Seiler, H. Kurniawati, and S. P. N. Singh, "An online and approximate solver for POMDPs with continuous action space," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2015, pp. 2290–2297.
[11] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 66–73.
[12] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3909–3917.
[13] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state," *Autonomous Robots*, May 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9746-1
[14] C. Dong, J. M. Dolan, and B. Litkouhi, "Interactive ramp merging planning in autonomous driving: Multi-Merging leading PGM (MML-PGM)," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (ITSC2017)*, Oct. 2017 ACCEPTED.
[15] C. Dong, Y. Zhang, and J. M. Dolan, "Lane-change social behavior generator for autonomous driving car by non-parametric regression in reproducing kernel hilbert space," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 4489–4494.
[16] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler, "Solution of the sylvester matrix equation axbt + cxdt = e," *ACM Trans. Math. Softw.*, vol. 18, no. 2, pp. 223–231, Jun. 1992. [Online]. Available: http://doi.acm.org/10.1145/146847.146929
[17] "NGSIM homepage. FHWA." 2005-2006. [Online]. Available: http://ngsim.fhwa.dot.gov.
[18] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami, "Conditional neural processes," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1704–1713.
[19] Y. Chen, "Learning-based lane following and changing behaviors for autonomous vehicle," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, May 2018.
[20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

Efficient Extrinsic Calibration System of a Camera and a 3D LiDAR with Accurate Feature Extraction and Error Diagnostics

Jiawei Tang[1]

Abstract— **The extrinsic calibration of a camera and a 3D Light Detection and Ranging(LiDAR) sensor is the precondition of combining the data in these two sensors for other processing. Most extrinsic calibration approaches need to detect the feature in a certain target, and their results are sensitive to the extracted features. This means the pose with an imprecise extracted feature can result in a large calibration error. In this paper, we develop a efficient extrinsic calibration system of a camera and 3D LiDAR by a checkerboard. We increase the calibration accuracy by improving the precision of feature extraction and improve the the robustness of an extrinsic calibration system with an error diagnostics approach in extracted features. This system is evaluated in real environment. The accuracy and reliability are strongly verified by the testing results. Meanwhile, in order to improve the usability of the whole system, we provide an efficient Matlab toolbox with an user-friendly GUI, which could benefit the large-scale industrial application.**

I. INTRODUCTION

In recent robotic research and applications, a camera and a 3D Light Detection and Ranging(LiDAR) sensor are often used together to collect environmental information. The color and texture of different objects could be easily captured through a camera. However, after transformation from world coordinate to the pixel coordinate by camera project, the deep depth information could not be obtained in the image [1]. While the 3D LiDAR could provide the complement of the deep depth by measuring the time-of-flight of the laser light. Based on the characteristics of the camera and the 3D LiDAR, they are currently widely used together in the fields of robotics.

In order to effectively combine the information obtained from both camera and 3D LiDAR, the extrinsic parameters, which include rotation and translation, between two sensors are required to obtain in advance. In the literature, the extrinsic calibration problem could be solved by a target-based method or a non-target-based method. In [2], [3], [4], [5], they obtain the extrinsic parameters through the mutual information, which includes the the LiDAR's reflectance and image's color. These methods provide a calibration system without using a particular target. However, comparing with the target-based method, the non-target-based method requires large number of poses to increase the calibration accuracy, which makes the calibration task more difficult. In the target-based approach[6], [7], [8], [9], a common target, like a checkerboard, or a special-designed target is used to estimate the extrinsic parameters. The special-designed target need to use special material or technology to satisfy their special geometric constraint, this special design increases the cost. As a common target, the checkerboard is widely used in many calibration systems, as it can provide lines and planes correspondence in the camera frame and LiDAR frame. Due to the features of the checkerboard needs to be extracted, the precision of the extracted features will influence the accuracy of the extrinsic calibration result as well as the robustness of the whole calibration system. In this situation, a precise feature extraction method and an efficient error diagnostic approach need to be provided for increasing the accuracy of the calibration result and the robustness of the calibration system.

When a large number of cameras and LiDAR need to be calibrated, the users do not hope to separate the calibration task to several subtasks. Meanwhile, they do not expect a complicated and cumbersome test setup, which may cost too much time or calibration failure. So an easy-to-used calibration toolbox should be proposed for solving these drawbacks.

In this paper, we present an efficient camera and 3D LiDAR extrinsic calibration system. We increase the calibration accuracy by making extracted features more precise in camera and 3D LiDAR. And we propose an error diagnostics approach for the extracted feature which could improve the robustness of the whole extrinsic calibration system. Finally, we provide a Matlab toolbox of the whole calibration system with an easy-used graphic user interface, which could benefit the large-scale industrial application.

The remainder parts of this paper are structured as follows: In Section II, we enumerate the related researches on the extrinsic calibration of the visual system and the range sensor. In section III, we describe the whole extrinsic calibration system of a camera and a 3D LiDAR. The feature extracting method as well as the error diagnostics approach would be described in the meantime. The experiment result of our extrinsic calibration system is presented in Section IV. Conclusion and future works are finally discussed in Section V.

II. RELATED WORK

The target-based approaches are widely used in the extrinsic calibration between the different visual system and range sensors. Gomez and Briales[10] propose an extrinsic calibration approach for a camera and a 2D laser-rangefinder based on orthogonal tetrahedrons on the corners. They estimate the extrinsic parameters based on the line-to-plane

[1]J. Tang is an Electronic and Information Engineering student at the Hong Kong Polytechnic University. This work was supported by Robotics Institute Summer Scholars program at Carnegie Mellon University .

and point-to-plane constraints obtained from the orthogonal tetrahedrons. Gong *et. al.* [11] solve the extrinsic calibration problem of 3D LiDAR and camera through the geometric constraints associated with a trihedral object. Through these methods could estimate the extrinsic parameters of the visual system and range sensors through the existing human-made object, it is incontinent to find a fine target in most situation, which means the usability of those approaches have huge limitation.

The checkerboard is treated as a common target for calibrating the extrinsic parameters between visual system and range sensors in many different approaches. Zhang and Pless [12] make use of plane-line correspondence in a planar checkerboard to perform the extrinsic calibration between a camera and laser rangefinder. This is the first extrinsic calibration approach for a camera and a laser rangefinder by using a checkerboard. In [13], [7], [8], they use the checkerboard to calibrate a camera and a 3D LiDAR. As they only consider the plane correspondence of the checkerboard, at least 3 poses need to be used for estimating the extrinsic parameters. This drawback has been solved by Zhou *et. al.* [9] with combining 3D line and plane correspondences. All of the methods mentioned above are needed to detect the features from the calibration target and sensitive to the precision of the features.

As a solution to deal with the real-world sensor system, a calibration tool could help users to utilize it efficiently. Unnikrishnan [14] provides a fast extrinsic calibration tool for a rangefinder and a camera. However, before using that tool, the intrinsic parameters need to obtain from another toolbox, Camera Calibration Toolbox for Matlab, which decreasing the tool's usability. Another toolbox used for calibrating the cameras and range sensor is proposed by Geiger *et. al.*[6]. This toolbox could calibrate different cameras and range sensor by one shot, however, as it needs to set up a large number of checkerboards at the same time, the robustness and usability of this toolbox are also limited.

III. SYSTEM DESCRIPTION

In this session, we firstly come up with an extrinsic calibration system of a camera and a 3D LiDAR using a checkerboard as the calibration target, which is shown in Fig. 1. The whole system includes features extraction in both camera and 3D LiDAR with data error diagnostics approach for extracted features and the extrinsic calibration and nonlinear optimization based on the line and plane correspondences for obtaining the extrinsic parameters between the camera and the 3D LiDAR. In order to benefit the large-scale industrial application, an end-to-end Matlab pipeline with user-friendly GUI is presented in the final part of this session.

A. Feature Extraction

The detected boundaries in the image and the 3D information of the checkerboard in laser point cloud are shown in Fig. 2.

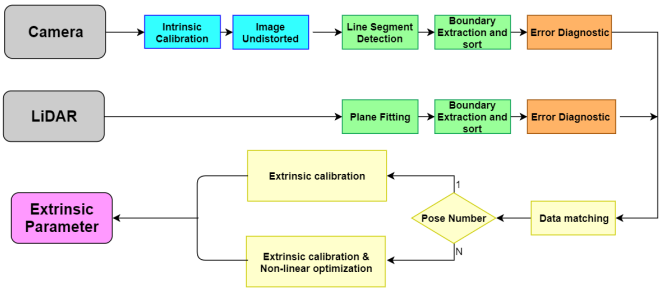


Fig. 1: Overview of the whole extrinsic calibration system, the system includes the front-end feature extraction in both camera and 3D LiDAR, and the back-end extrinsic calibration.

The image captured by a pinhole camera is distorted by the camera distortion parameters. In order to reduce the influence of the distortion for the line detection in the image, the intrinsic parameters should be obtained in advance. We use Zhang's method[15] to calculate the intrinsic parameters of the camera and recover the original image to undistorted image.

For the feature in the image, We use LSD algorithm [16] to detect the lines in the undistorted image. We optimize the LSD algorithm when implementing it in Matlab. We smooth the image by performing Gaussian filtering using convolution to eliminate the noise in the background, and we fuse broken line segments in the same line to which results from the change of light intensity. As we can get the four apical corners from the intrinsic calibration, the 4 boundaries which enclose the checkerboard could be detected in the group of extracted lines. In camera, the plane and boundaries in *jth* image are donated as π_{ji} and $\boldsymbol{l}_{ji}$, where i = 1, ,2 ,3 ,4 means the *ith* boundary. The direction of the checkerboard's plane in the image and the distance between the original and the plane in are donated as $\mathbf{n}_j$ and $\mathbf{x}_j$. The direction of the checkerboard's boundary in the image and a point in the boundary are donated as $\mathbf{d}_{ji}$ and $\mathbf{y}_{ji}$.

As for the checkerboard in the LiDAR point cloud, we firstly provide a particular cuboid which could enclose the whole checkerboard. Inside that area, we detect the laser points on the checkerboard by using RANSAC algorithm[17] as the detected plane. In this plane, we fit the fringe laser points as the checkerboard boundaries. The plane and boundaries of the checkerboard in *jth* point cloud are donated as $\prod_{ji}$ and $\boldsymbol{L}_{ji}$, where i = 1, ,2 ,3 ,4 means the *ith* boundary. The normal of the checkerboard plane in the point cloud and a point in the boundary are donated as $\mathbf{N}_j$ and $\mathbf{X}_{ji}$. The direction of the checkerboard boundaries in the point cloud and a point in the boundary are donated as $\mathbf{D}_{ji}$ and$\mathbf{Y}_{ji}$.

B. Error Diagnostics for Extracted Features

Here we verify the correction of detected lines based on the homographic transform of lines in the image and the perpendicular constraint of the coterminous detected checkerboard boundaries in laser point cloud.

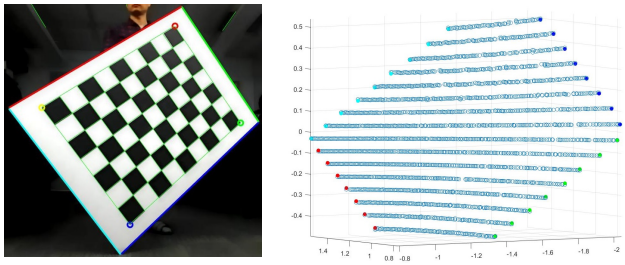


Fig. 2: The extracted features, including the lines and plane, of a checkerboard in the camera and 3D LiDAR.

A point, x, in a line, L, satisfies the following geometric constraint:

$$L^T \cdot x = 0 \tag{1}$$

where $\boldsymbol{L}^T$ is the transposition of the line and $\boldsymbol{x}$ is a point inside $\boldsymbol{L}$. As for a pair of corresponding points in pixel coordinate, $\boldsymbol{x}_W$, and in world coordinate, $\boldsymbol{x}_I$, they satisfies that:

$$x_W = H \cdot x_I \tag{2}$$

where, H is the homographic matrix from pixel coordinate to world coordinate. Based on the geometric constraint in (1) and (2), the congruent relationship of the corresponding lines of the checkerboard in image coordinate, $\boldsymbol{L}_I$, and that in world coordinate, $\boldsymbol{L}_W$, could be noted as:

$$L_W = H^T \cdot L_I \tag{3}$$

This shows that all of the corresponding lines of the same planar surface in space could be transformed to each other through the homographic matrix. As the Homographic matrix could be estimated by four corresponding points in the checkerboard [1], we could diagnose each detected boundaries in the image by projecting them into the checkerboard respectively. We set angle threshold, θ, to estimate whether the boundary we detected is the correct one in the checkerboard based on the angle between the original boundary and transformed boundary.

The coterminous boundaries of the checkerboard should satisfy the perpendicular constrain as they constitute a rectangular. For the *ith* boundary detected in the jth LiDAR scan, we denoted its direction as $\boldsymbol{d}_{ji}{}^P$, where i = 1, ,2 ,3 ,4. As the points on the boundary exit slight noise, we set a threshold, ν, to determine the perpendicular constrain of detected boundaries:

$$d_{ji} \cdot d_{ji+1} < \nu \tag{4}$$

We diagnose the correction of the detected lines in image and laser point cloud by (3) and (4). As the result of extrinsic calibration could be obtained with only one pose by using the method presented in session C, we discard the pose which the detected boundaries could not satisfy line-to-line homographic transformation in image and perpendicular constrain in point cloud at the same time. The precisely detected features are used in the extrinsic calibration system can reduce the calibration failure due to the bad feature extraction.

C. Extrinsic Calibration

We solve the extrinsic calibration problem between the camera and the 3D LiDAR based on 3D line and plane correspondences of the checkerboard. One correspondence lines in camera and 3D LiDAR could provide four independent constrains on rotation, $\boldsymbol{R}$, and translation, $\boldsymbol{t}$, from LiDAR to camera, and one correspondence planes provides three independent constraints on $\boldsymbol{R}$ and $\boldsymbol{t}$. The constraints provided by 3D line and plane are shown as follow:

$$RD_{ji} = dji \tag{5}$$

$$(I - d_{ji}(d_{ji}^T))(RY_ji - x_ji + t) = 0_{3\times1} \tag{6}$$

$$R \cdot n_{ji} = N_{ji} \tag{7}$$

$$n_j \cdot (RX_{ji} + t) + d_j = 0 \tag{8}$$

As there are four perpendiculars and one plane in a checkerboard, these constraints guarantee that the extrinsic parameters between the camera and 3D LiDAR could be estimated with only one pose. As all of the detected boundaries in image and point cloud should be matched with each other, we put the detected boundaries in one pose to a RANSAC framework. We first sort the detected boundaries based on the order of connection in image and LiDAR. As the sorted boundaries exit four possible matching relationships, we calculate the R and t four times to get four results. Then we project the boundaries in LiDAR to the camera using R and t and calculate the residual error of the corresponding boundaries. As only one result provides the correct matching relationship, we treat the result with minimum residual error as the final one pose solution of our calibration problem. As the noise in point cloud is non-negligible, we reduce the influence of the noise by increasing the number of checkerboard's poses and solve the nonlinear optimization problem by using Levenberg-Marquardt method[18].

D. Visualized Matlab User Interface

The Matlab GUI, shown in Fig. 3, provides a direct solution for helping the user obtain the extrinsic parameters rapidly and easily. As we mentioned in Session A, a particular 3D area, which includes all of the laser points on the checkerboard, should be figured out in the point cloud. The function of Test XYBoundary provides a visualized 3D graph for helping the user figure out the area of the checkerboard, which could reduce calibration failure by human mistake. By simply clicking the Extrinsic Calibration button, after providing the data path, the number of data, square size of the checkerboard and X,Y boundary, the user could observe the operation of the whole calibration process in the GUI with visualized graphs and get the final calibration result.

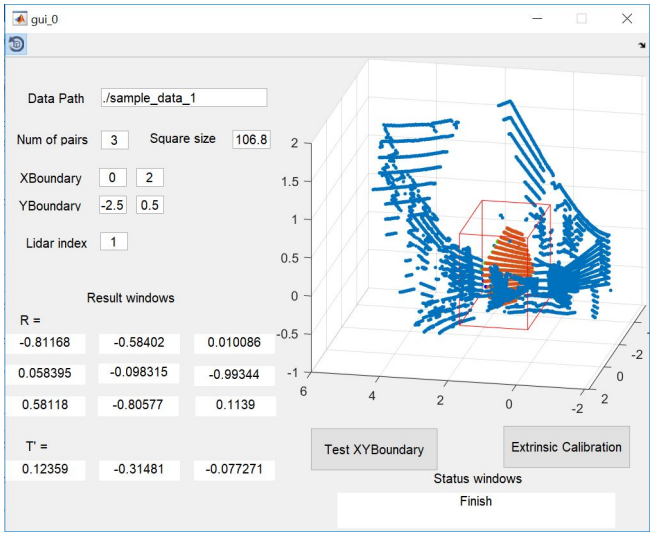


Fig. 3: The end-to-end Matlab pipeline with easy-used GUI for the whole extrinsic calibration system. The figure in the GUI is the visualized point cloud and a red cuboid figured out by the user for extracting the laser point on the checkerboard.

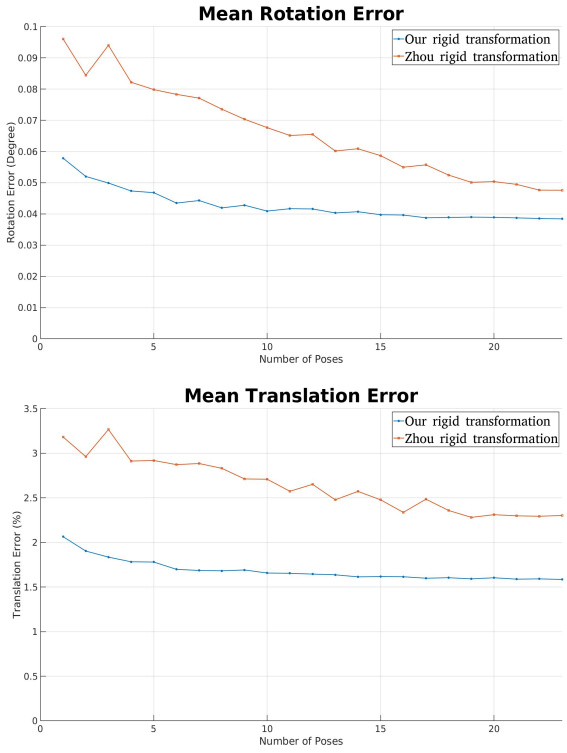


Fig. 4: The rotation error and translation error of our method and Zhou's method [9]. The calibration accuracy has significant improvement with a more precise extracted features in image and 3D LiDAR.

IV. EXPERIMENTS RESULT

For testing our extrinsic calibration system, we use a Velodyne VLP-16 LiDAR and a ZED stereo camera as our test platform. In accuracy evaluation, We use 32 pairs of LiDAR point cloud and images as our calibration data. This dataset is collected and used in [9]. The orientation of the checkerboard in this dataset is well-distributed. And it was collected in a simple environment with all of the checkerboard's boundaries could be extracted in both image and point cloud. We use this dataset to verify the accuracy of the extrinsic calibration system. In robustness evaluation, we collect data in different light condition and increase the noise in the point cloud by keeping moving the checkerboard, We verify the performance of our error diagnostics approach by observing whether it could provide an alert when the extracted data is wrong.

A. Accuracy Evaluation

To evaluate the accuracy of our extrinsic calibration system, we use the extrinsic parameters of the stereo camera ($\mathbf{R}_0$, $\mathbf{t}_0$) as the ground truth. We first estimate the extrinsic parameters of the LiDAR with the left ($\mathbf{R}_1$, $\mathbf{t}_1$) and right camera ($\mathbf{R}_2$, $\mathbf{t}_2$), then we estimate the extrinsic parameters of the stereo camera ($\hat{\mathbf{R}}$, $\hat{\mathbf{t}}$) through ($\mathbf{R}_1$, $\mathbf{t}_1$) and ($\mathbf{R}_2$, $\mathbf{t}_2$). We calculate the rotation angle of $\hat{\mathbf{R}}\mathbf{R}_0$ in the angle-axis representation[19], which is used to evaluate the rotation error and the translation error by $\|\hat{\mathbf{t}} - \mathbf{t}_0\|_2 / \|\mathbf{t}_0\|_2$.

We compare rotation error and translation error of our calibration result with that getting from Zhou's result. For one pose result, we estimate the rotation error and translation error for all the 32 pairs of LiDAR point cloud and images. For N poses result, we choose the data pairs from the dataset randomly and run the experiment 200 times. As the result shown in Fig. 4, our system only has 0.058 degree rotation error and 2.1% translation error in one pose result, which could be achieved in Zhou's approach in 15 poses result. This indicates that our optimization in feature extraction could provide a significant improvement in the extrinsic calibration system.

B. Robustness Evaluation

We collect 50 checkerboard poses from the camera and 3D LiDAR in different light condition, to evaluate the robustness of our calibration system. We increase the noise of the laser point on the checkerboard's boundaries by keeping moving the checkerboard when collecting the data.

The result shows that in the extremely bad light condition, the boundaries of the checkerboard cannot be detected successfully due to the change of gradient on the boundaries is too small And the fitting lines of the checkerboard's boundaries is incorrect if the movement is too strong when collecting data. In the 50 point cloud and image pairs, 5 images could not provide correct checkerboard boundaries and 15 point cloud could not provide the correct fitting lines of checkerboard boundaries. These incorrect poses are detected from our error diagnostics approach.

V. CONLUSION AND FUTURE WORKS

In this paper, we develop an efficient calibration system of a camera and a 3D LiDAR. We improve the accuracy of the calibration result by improving the precision of feature

extraction and make our more robust system by an error diagnostics approach in extracted features. Experimental results show that our system can provide an accurate calibration result with only 0.058 degree rotation error and 2.1% translation error, this is a significant improvement compared to the current start-of-the-art method. We also verify the robustness by using a set of noisy data. We also improve the usability of our system by developing an easy-to-use Matlab toolbox with a user-friend GUI, this makes our system can be utilized in the large-scale industrial application efficiently.

In future works, as our approach just uses the line and plane correspondence for calibration, we will consider applying our extrinsic calibration approach to an online camera and LiDAR localization and mapping system. We are also interested in applying this approach to a multi-camera and multi-LiDAR system, Which can benefit the application in autonomous driving or other robotic application.

VI. ACKNOWLEDGE

This work has been supported by the Robotics Institute for Summer Scholars and the Robot Perception Lab at Carnegie Mellon University. We would like to thank Prof. Michael Kaess, Dr. Lipu Zhou, and Mr. Zimo Li for helping me finish this work. Specially thank the Hong Kong Polytechnic University for providing UG Summer Research Abroad Sponsorship for funding me in this summer program. And deeply thank Ms. Rachel Burcin, Dr. John Dolan, Mr. Ziqi Guo, and the rest of RISS for a great summer.

REFERENCES

[1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: Science and Systems*, vol. 2, 2013.

[3] Z. Taylor and J. Nieto, "Automatic calibration of lidar and camera images using normalized mutual information," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013.

[4] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Choosing a time and place for calibration of lidar-camera systems," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4349–4356.

[5] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information." in *AAAI*, 2012.

[6] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "A toolbox for automatic calibration of range and camera sensors using a single shot," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2012.

[7] L. Zhou and Z. Deng, "Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 642–648.

[8] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, "3d lidar–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 452–467, 2012.

[9] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences," in *Intelligent Robots and Systems, 2018. IROS 2018. IEEE/RSJ International Conference on*. IEEE, 2018.

[10] R. Gomez-Ojeda, J. Briales, E. Fernandez-Moral, and J. Gonzalez-Jimenez, "Extrinsic calibration of a 2d laser-rangefinder and a camera based on scene corners," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3611–3616.

[11] X. Gong, Y. Lin, and J. Liu, "3d lidar-camera extrinsic calibration using an arbitrary trihedron," *Sensors*, vol. 13, no. 2, pp. 1902–1918, 2013.

[12] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)." in *IROS*, vol. 3, 2004, pp. 2301–2306.

[13] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic calibration of a 3d laser scanner and an omnidirectional camera," in *7th IFAC symposium on intelligent autonomous vehicles*, vol. 7, no. 1, 2010.

[14] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09, July 2005.

[15] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.

[16] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a Line Segment Detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[18] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[19] Z. Kukelova, J. Heller, and A. Fitzgibbon, "Efficient intersection of three quadrics and applications in computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1799–1808.

Visually Descriptive Image Captions: Improve Your Model With No Additional Training

Brandon Trabucco*, Junjiao Tian, Roberto Shu, Jean Oh, Ralph Hollis
Department of EECS; Robotics Institute*
University of California, Berkeley; Carnegie Mellon University*
Berkeley, CA*; Pittsburgh, PA
btrabucco@berkeley.edu*, { junjiaot, rshu, hyaejino, rhollis }@andrew.cmu.edu

***Abstract*—Image captioning receives significant attention by the deep learning research community, as scientists maintain large collections of public datasets and rapidly invent benchmarks and algorithms. Existing image captioning frameworks learn to generate sentences directly from ground truth pairs of images and captions. However, previous methods commonly produce ambiguous descriptions for two images with subtle differences, which prevents the adoption of image captioning systems in real world settings. We propose a captioning algorithm that reduces this problem, and produces more visually descriptive image captions. We develop a novel reinforcement learning cost function that encourages generating captions with stronger word choice, and a novel inference adaptation method that allows our heuristic to improve the performance of existing image caption frameworks using style transfer. Our work is being tested for deployment on a mobile robot that navigates through an office and lab building.**

I. Introduction

The Artificial Intelligence community is currently witnessing an explosion of interest in multiple discipline research. In particular, research at the intersections of computer vision, natural language processing and robotics is dramatically increasing. Within this movement, we observe researchers are defining new problems with accompanying datasets—we focus on Image Captioning and MSCOCO, but many other examples exist [1]–[3].

These datasets are encouraging an entirely new class of deep learning frameworks leveraging convolutions for images, recurrence for language, and memory [4]–[10]. These benchmarks and algorithms are citing a belief: that combining multiple data modalities (eg: image and text) creates a stronger relational understanding of the real world, and that such understanding is crucial for developing universal and general artificial intelligence [1], [2].

This belief is motivating researchers to investigate Image Captioning. These researchers are developing expressive neural networks capable of modeling fine interactions between visual features and sentence words [4]–[6], [9]–[12]. Indeed, these architectures are consistently improving with respect to common evaluation metrics such as BLEU-4 score [6].

One of the primary goals motivating these inventions is to learn captions indistinguishable from human annotated ground truth examples [4], [6], [10]. This method works quite well according to multiple common evaluation metrics [13], [14]. Because of these results, interest within the Image Captioning research community is beginning to shift away from the fundamental problem definition into more specific domain applications—for example, captioning novel objects, and word style [5], [7], [8], [11].

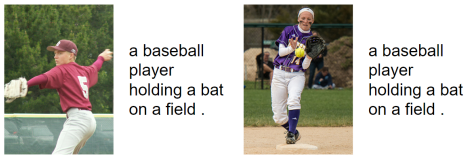


Fig. 1. Captions produced by [4] are ambiguous. In the left image, a pitcher is throwing a baseball. In the right image, a softball player is catching a softball. The baseline algorithm produces the same caption for both images.

These recent approaches introduce new capabilities that increase the utility of Image Captioning systems, which motivate the question of whether these systems can be integrated with humans. We envision a robot platform that integrates Image Captioning with a camera and speech interface, where humans verbally command the robot to describe the surrounding environment. This application requires captions produced by the model to be close to error free, and descriptive enough that phrases can be visually identified.

However, these requirements are difficult to satisfy with existing algorithms. We depict in Fig. 1 that our baseline frequently misses contextual information within images that requires abstract reasoning [7], [8]. Furthermore, a lack of descriptive language requires explicit model adaptation to correct [5]. Therefore, we conclude that before an Image Captioning system can be deployed with humans, new algorithms for descriptive captions are required.

In this paper, we expect to make the following specific contributions to Image Captioning research.

- The first robot integrated Image Captioning system descriptive enough to be useful to humans [1].
- Two novel cost function that encourage use of descriptive words and visual words.
- An novel style transfer algorithm for making captions

[1]Code for our robot implementation can be found at http:// github.com / brandontrabucco / image_caption_machine

descriptive [2].

II. Related Work

Image Captioning is a sequence generation problem, here a sentence is generated using an image as input. Deep neural networks are the standard method by which to extract visual image features, and to model the complex interactions between words in a sentence. The image of width X, height Y, and colors C is transformed by ϕ parameterized by θ into a sequence of integer word ids y_i in a vocabulary.

$$\phi_\theta : \mathbb{R}^{X \times Y \times C} \rightarrow \{[y_0, y_1, \ldots, y_N] : \forall y_i \in \mathbb{Z}\}$$

The first publication reporting major success in training an end-to-end deep neural network for image captioning was [4], where an image encoder similar to [15] was used to compute global image features, and a sequence decoder similar to [16] was used to compute word probabilities at every time step.

A. Quality Of Word Embeddings

Collections of papers including [4], [6], [10] learned word embeddings directly from captions, with no preemptive training. Other papers including [7], [8] use various word embeddings trained on Wikipedia [17], [18]. Using high quality word embeddings appears optional to obtain strong testing performance on caption datasets.

Somewhat to be expected, models without high quality word embeddings over fit to their source datasets, and fail to learn concepts seen infrequently in training [7], [8], [11]. Certain approaches leverage external text and image datasets to learn these unfamiliar objects [7], [8]. Other methods achieve the same capabilities by forcing latent image features to be attributes [11].

Explicitly learning captions conditioned on image attributes rather than latent image features appears to generally improve performance [11], [19], [20]. Reflected in these papers, it appears high quality word embeddings are required to generalize beyond the source dataset.

B. Producing More Detailed Captions

Many traditional works for Image Captioning propose novel additions to [4] that must be trained and deployed independently [4], [6], [9]–[11], [19]. Adapting these models during inference is briefly investigated by [5], which produces more descriptive captions that previous work. [5], [21], [22] continue and augment [4] to learn stylistic captions during training.

Multiple of these approaches require optimizing a reinforcement learning objective, which [5], [23], [24] demonstrate to be effective. From these works, obtaining domain specialized captions appears to require significant changes to the underlying model architecture or training scheme.

One relevant contribution inspiring our method is [5], where they invent a critic-based planning algorithm by substituting actions that maximize some Q-function during inference. Their method requires a significant amount of unaligned text for training auxiliary critics on both source and target distributions. Our approach differs from theirs in that our inference adaptation algorithm does not directly insert words into the caption, rather we move through the language prior directly using neural style transfer.

III. Methods

Our baseline model is the [4]. We select this because numerous scientific evaluations of this model already exist, and many recent Image Captioning algorithms also use this as a baseline. We begin with a public repository [3] published by the authors of [25], which can be summarized by a $\mathbf{CNN}$ encoder that maps an image I into visual features, on which we take the mean across dimensions $[X, Y]$, and map into an embedding space W_{image} to produce a spatially invariant feature context vector $\vec{c}$.

$$\vec{c} = W_{image} mean(\mathbf{CNN}(I), [X, Y])$$

We use this context vector $\vec{c}$ in the same manner as [25], where the $\mathbf{LSTM}$ reads $\vec{c}$ mapped into an image embedding space and generates an initial state $\vec{s}_0$ characterizing the image contents. After this point, the LSTM no longer receives the image features directly as input.

$$\vec{s}_0 = state(\mathbf{LSTM}(\vec{c}, \vec{0}))$$

At every decoding timestep, the $\mathbf{LSTM}$ produces a set of word probabilities, which are sampled to produce the current word identifier y_t. The previous word that was sampled is embedded $\eta(y_{t-1})$ using word vectors from [17], and is input to the $\mathbf{LSTM}$ along with the previous state $\vec{s}_{t-1}$. In the first timestep, the first $\vec{s}_0$ state is conditioned only on the image. V^T_{glove} is initialized with the transpose of word vectors from [17]. V^T_{glove} is trainable, while the embedding function η is fixed.

$$P(y_t) = V^T_{glove} output(\mathbf{LSTM}(\eta(y_{t-1}), \vec{s}_{t-1}))$$

$$y_t \sim P(y_t)$$

We select the [17] embedding space because prior work [7], [8], [11], [19], [20] demonstrates that high quality word embedding are a requirement for producing captions significantly different from those seen in training. Our goal is to generate more visually descriptive captions than what was seen in training, and it follows to use [17].

Research from [7], [8] also demonstrates that training the $\mathbf{LSTM}$ using an external language corpus such as Wikipedia successfully enables captioning of objects that were never seen before. This is a useful property for our purposes, so we jointly train our language model on a subset Wikipedia.

[2]Code for our tensorflow implementation can be found at http:// github.com / brandontrabucco / im2txt

[3]The original implementation can be found at https:// github.com / tensorflow / models / tree / master / research / im2txt

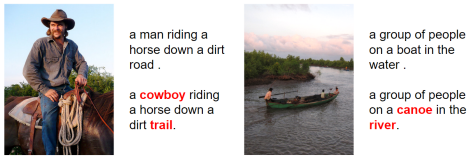


Fig. 2. Captions produced by [4] use ambiguous words. For each image, the top caption is produces by the baseline algorithm, and the bottom caption depicts one possible way to replace words to add more detail.

We train the baseline model [4] using supervision from pairs of images and ground truth captions. After the decoding phase, the **LSTM** has produced a sequence of word identifier y_t, and we are given the ground truth words $\hat{y}_t$. We use the standard mean cross entropy loss with a vocabulary of N words and a caption of length L. The probabilities $P_j(y_j)$ below represent the jth possible word at the ith time.

$$\delta(\theta) = \frac{-1}{LN} \sum_{i=0}^{L} \sum_{j=0}^{N} P_j(\hat{y}_i) \log P_j(y_i)$$

We train our model using the previous loss function on the [1] dataset. We perform stochastic gradient descent with a batch size of 32, and no additional regularization on the weights and biases in the model. The **CNN** internally computes batch normalization [15], and we apply dropout to the **LSTM** during training only with a keep probability of 0.7. The embedding space has 300 dimensions, and so does the state vector of the **LSTM**. The vocabulary has the 70,000 most frequent words from a 6 billion word corpus [17].

A. Descriptive Cost Function

We initially observed that certain words were preferred by our baseline model when generating captions. In these instances, more descriptive alternative words exist. For example, replacing the word ”man” with ”cowboy” as in Fig. 2. We believe that performing such a replacement during inference time will generate captions with more descriptive details.

We define a measure for the number of synonyms a word has as the number of close neighbors in the embedding space from [17] within some threshold. This is logical because it is known that the distance of words in the embedding space from [17] implies these words have similar definitions.

To calculate our measure for the number of synonyms a word has, we look at a set of common words with known synonyms in Fig. 3. We take these words and calculate an average distance from a synonym to an origin word. This distance is a threshold for if words are synonyms.

Equipped with this metric, we calculate the average number of synonyms the words in some collection have. We believe that more descriptive words tend to have fewer close synonyms. We compute the average number of synonyms of words from the entire model vocabulary [17], from the captions produced by [4], and from the ground truth captions [1].

Word	Synonyms
man	woman, person, another, one, he, boy, himself, him
walk	walking, walks, walked, go going, stroll, come
path	paths, way, direction, heading approach, trajectory, means, turn
fruit	fruits, citrus, berries, mango bananas, apples, banana, watermelon strawberries, grape
building	buildings, built, houses, construction constructed, build, addition, structure constructing, structures, tower, offices

Fig. 3. Words and synonyms used to calculate the mean synonym distance for embeddings from [17]. On the left is the origin word, and the right are related words selected from the 20 closest words in embedding space.

Distance Threshold
5.632423353

Fig. 4. The mean distance from the origin word to the synonym words in the [17] vocabulary was calculated, which will serve as a threshold for calculating how many synonyms a given word has.

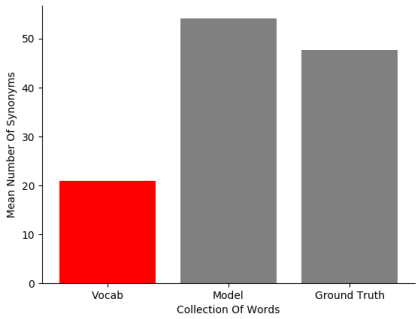


Fig. 5. The mean number of synonyms in the [17] vocabulary, shown in red, is less than those learned by the [4] model, shown in the middle, which is higher also than the number of synonyms of the ground truth captions, shown to the right.

Our results in Fig. 5 confirm the initial observation that the baseline model resorts to using more general words when more descriptive synonyms are available. We also learn that the original dataset also resorts to more general words choice. We propose to combat this by explicitly encouraging the model to use words that have fewer synonyms using a novel descriptive cost function.

In order to score words with less synonyms in the embedding space higher, take the sum of distances from word x to the K closest other words y in the embedding space.

$$D(x) = \sum_{i=0}^{K} \min_{y \neq x}[i] ||\eta(x) - \eta(y)||$$

Even though we employ this cost function with our method for style transfer, this cost function may even be used in

training as regularization, but we do not present results for this application.

B. Visual Cost Function

Recent publications demonstrate that detected attributes can improve the performance of image captioners according to dataset evaluation benchmarks [11], [19], [20]. We observe in [11], [19] that inclusion of attributes changes the style of captions produced by [4] to more descriptive.

A natural extension of previous work for captioning with attributes should focus on leveraging training data from other target domains. For example, two rich attribute datasets are [26], [27]. Another natural consideration is to frame detection of attributes instead as descriptive words. This provides access to a larger set of attributes to detected, and integrates easily with our method of neural style transfer.

Therefore, we build an attribute detector using [15] as a base model. Our detector takes an image I as input, along with a word x from our vocabulary. Our detector discriminates the probability that word x is visually grounded in image I. $\vec{c}$ is the image context vector produced by [15], and $\eta(x)$ is the embedding vector of word x using [17]. We take a linear combination of the embeddings and scale the result by a logistic function to obtain a probability.

$$G(x, I) = \sigma(W_{attr}(\vec{c} \odot \eta(x)))$$

We train this attribute detector using [26], where each attribute has been mapped to a unique relevant word in our vocabulary. Since our cost function returns probabilities, not attributes, we can obtain the K most relevant attributes by collecting those with the largest probabilities.

$$A = \{\arg\max_x[i] G(x, I) : i \in [0, K]\}$$

Similar to our previous cost function, we employ this cost function with our method for style transfer. However, this cost function may even be used in training as regularization, but we do not present results for this application.

C. Inference Adaptation

Previous work by [5] demonstrates that changing the output of [4] during inference is an effective strategy to change the phrasing and style of captions. We notice their method relies on adapting the model during a secondary training phase, and additionally during inference to produce high quality results.

We are curious if an inference only adaptation method could also produce equally descriptive and high quality image captions. The benefit of such a method is that no additional training of an image captioner would be required, and a pretrained model could be downloaded from the internet, and styled to match our needs.

One setback of the method proposed in [5] is that changes to the model outputs during inference operate on the word level. Rich information contained in the model's language prior is not used to globally plan when and where to insert words. This is a problem we correct in our method.

#	**Algorithm**
0	def caption():
1	Encode, Decode = load ("model.ckpt")
2	s_0 = Encode ("image.jpg")
3	for x in range(N):
4	y_0 = id ("$< S >$")
5	$y_1, \ldots, y_L$ = Decode (y_0, s_0)
6	$\tilde{Q} = \sum_{i=1}^{L} P(y_i)D(y_i) + P(y_i)G(y_i, I)$
7	$s_0 \leftarrow s_0 + \gamma \nabla_{s_0} \tilde{Q}$
8	return word ($y_1, \ldots, y_L$)

Fig. 6. We load the [15] image encoder and an LSTM sequence decoder trained on [1] for 100,000 iterations with batch size 32. We calculate the beam search $y_1, \ldots, y_L$ at every iteration, along with the expected value $\tilde{Q}$. We perform gradient ascent $s_0 + \gamma \nabla_{s_0} \tilde{Q}$ once per iteration.

In previous work on text style transfer, summarized by [28], [29], a decoding RNN is trained generate sentences that match the style of a target domain. In some cases, an adversarial critic is used to numerically quantify style. In both settings, explicit training of the styled language generation model is necessary. We demonstrate a style transfer method that requires no additional training of a language model.

We define an objective Q that is the sum of the descriptive score $D(y_i)$ and visual grounding probability $P(y_i|I)$ for every word y_i in a caption produced by [4].

$$Q = \sum_{i=0}^{L} D(y_i) + G(y_i, I)$$

In order to perform a sort of global planning using the language prior of the RNN, we wish to maximize the value of Q with respect to the initial state s_0 of the RNN. By doing this, the language prior still controls the structure and word choice of the resulting caption, while certain visual features of the initial state s_0 are being made stronger.

Unfortunately, Q is a discrete function and $\nabla_{s_0} Q$ is not defined, so we must instead maximize the expected value for Q. which we can reformulate using the policy gradient theorem into a defined gradient.

$$\mathbb{E}[Q] = \sum_{i=0}^{L} \mathbb{E}_{y_i}[D(y_i) + G(y_i, I)]$$

$$\nabla_{s_0} \mathbb{E}[Q] = \sum_{i=0}^{L} \mathbb{E}_{y_i}[\nabla_{s_0} \log P(y_i)(D(y_i) + G(y_i, I))]$$

We perform gradient ascent on the expected value $\mathbb{E}[Q]$ with respect to the initial state s_0 with a learning rate γ, which in practice we have found to be in the order of magnitude of thousands. After each inference step, we immediately computed an updated beam search for y_i. We precisely define our algorithm in Fig. 6.

$$s_0' = s_0 + \gamma \nabla_{s_0} \mathbb{E}[Q]$$

Name	BLEU-4	CIDEr
NIC	**27.7**	**85.5**
DetailedNIC	22.9	80.4

Fig. 7. We sample 320 images from the validation dataset, and compute the beam search captions for [4] with a beam size of 3. We then compute the stylistic transfer with a learning rate of 1000.0 and 20 iterations to obtain styled captions. For each set, we compute the BLEU-4 and CIDEr scores.

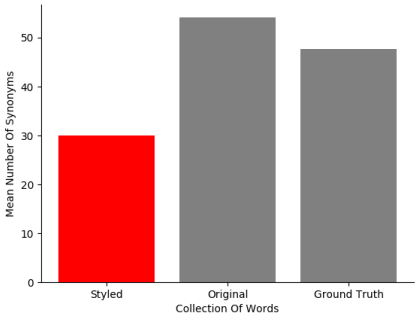


Fig. 8. The mean number of synonyms of words in the styled captions vocabulary, shown in red, is less than those learned by the [4] model, shown in the middle, which is higher also than the number of synonyms of the ground truth captions, shown to the right.

Our implementation of the policy gradient is vanilla, with no additional constraints or penalties to KL divergence, and no use of the fisher information matrix. Investigating the affect of these more recent approaches to the policy gradient is an important avenue for future work.

IV. Results

A. Benchmark Performance Trade Off

The first interesting result is that our method scores lower according to standard evaluation metrics on the validation dataset for [1]. According to Fig. 7, our method has somewhat of a large margin of performance loss when compared with [4]. Normally, this would imply that our method is inferior to existing approaches, but consider our findings in Fig. 5, that the original dataset commonly uses general words with many synonyms in the captions. Instead of using these common metrics to evaluate performance, perhaps a more enlightening indicator would be that from Fig. 5.

B. Our Method Is More Descriptive Than Baseline

Our results clearly depict in Fig. 8 that our styled model produces words with less synonyms than the original model, and also than the original dataset. We have been able to push the model to include more distinct words than what it was originally trained on, using only stylistic transfer. Also worth mentioning is that the [4] model has learned a word choice that has more synonyms er word on average that the original dataset. This reinforces our belief that learning to use more common words with more synonyms is actually a naive solution to learning image captions that achieves a high score using typical metrics.

C. Qualitative Results

Our results depict in Fig. 8 that our styled model produces words with less synonyms than the original model, and also than the original dataset. We have been able to push the model to include more distinct words than what it was originally trained on, using only stylistic transfer. Also worth mentioning is that the [4] model has learned a word choice that has more synonyms er word on average that the original dataset. This reinforces our belief that learning to use more common words with more synonyms is actually a naive solution to learning image captions that achieves a high score using typical metrics.

V. Discussion

Our results display an interesting trade-off between the ability to generate captions with fine visual details, versus captions that achieve high scores on benchmark datasets. In our experiments, these two classes typically do not share members. In order to build systems that perform accurately in the real world, like our implementation on a mobile robot, awareness of this trade-off is necessary, which prior work addresses in a more limited scope.

Datasets within the scientific community are not always treated with an appropriate level of skepticism. Supervision on these datasets—for example, to learn image captions—may cause models to learn implicit biases present in the data. We have shown from our experiments that the [1] dataset is biased towards image captions with word choice that has many synonyms (implying generality) and only superficial visual details.

Considerations of data bias are necessary when implementing real world robotic systems that leverage image captioning, especially where the application of the robot differs significantly from the domain of images from the original training dataset. Using auxiliary cost functions such as for descriptive word choice and attributes can reduce the affect of the original dataset bias, and may be useful in other domains of machine learning as a generalization strategy.

We are currently testing our image captioning framework on a Ballbot—a mobile robot that balances on a single spherical wheel [30]—to describe humans and objects in an office setting. Our hope is that research in more descriptive captions enables more effective human robot interactions in office settings, and eventually in health care and home settings.

VI. Acknowledgements

Thank you to Dr. Jean Oh and Dr. Ralph Hollis for their wisdom and helpful comments throughout this project. Special thanks to Rachel Burcin, John Dolan, Ziqi Guo, and the many other key organizers for the Robotics Institute Summer Scholars (RISS) program. Thank you to the Microdynamic Systems Lab (MSL) for providing a space to work on this project, and thank you also to RISS, MSL, the Carnegie Mellon University Robotics Institute, and the National Science Foundation grant IIS-1547143 for funding this project.

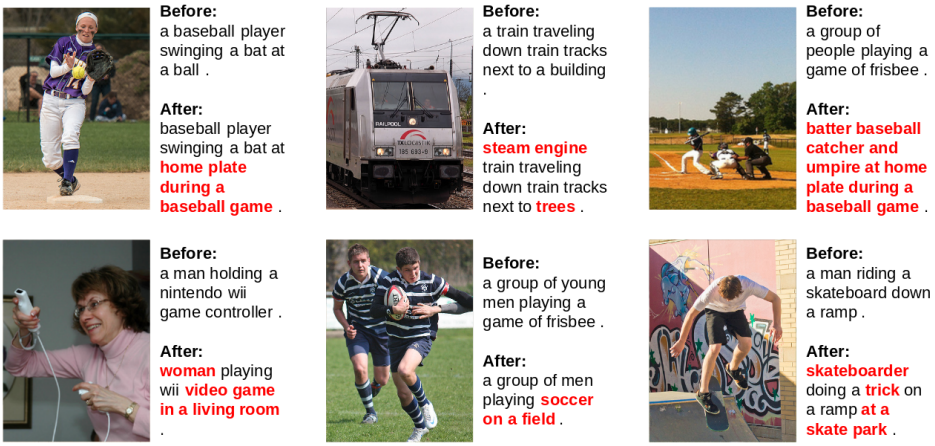


Fig. 9. We depict captions generated by the baseline [4] on the top half for each image, and captions produced by our descriptive stylistic transfer algorithm on the bottom. Key word differences between the methods have been highlighted in red.

REFERENCES

[1] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: visual question answering," *CoRR*, vol. abs/1505.00468, 2015.

[3] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," *CoRR*, vol. abs/1711.07280, 2017.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *CoRR*, vol. abs/1411.4555, 2014.

[5] T. Chen, Y. Liao, C. Chuang, W. T. Hsu, J. Fu, and M. Sun, "Show, adapt and tell: Adversarial training of cross-domain image captioner," *CoRR*, vol. abs/1705.00930, 2017.

[6] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via A visual sentinel for image captioning," *CoRR*, vol. abs/1612.01887, 2016.

[7] S. Venugopalan, L. A. Hendricks, M. Rohrbach, R. J. Mooney, T. Darrell, and K. Saenko, "Captioning images with diverse objects," *CoRR*, vol. abs/1606.07770, 2016.

[8] L. A. Hendricks, S. Venugopalan, M. Rohrbach, R. J. Mooney, K. Saenko, and T. Darrell, "Deep compositional captioning: Describing novel object categories without paired training data," *CoRR*, vol. abs/1511.05284, 2015.

[9] Z. Yang, Y. Zhang, S. ur Rehman, and Y. Huang, "Image captioning with object detection and localization," *CoRR*, vol. abs/1706.02430, 2017.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015.

[11] T. Yao, Y. Pan, Y. Li, and T. Mei, "Incorporating copying mechanism in image captioning for learning novel objects," *CoRR*, vol. abs/1708.05271, 2017.

[12] F. Faghri, D. J. Fleet, R. Kiros, and S. Fidler, "VSE++: improved visual-semantic embeddings," *CoRR*, vol. abs/1707.05612, 2017.

[13] R. Vedantam, C. L. Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," *CoRR*, vol. abs/1411.5726, 2014.

[14] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "SPICE: semantic propositional image caption evaluation," *CoRR*, vol. abs/1607.08822, 2016.

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[16] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, pp. 850–855 vol.2, 1999.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[19] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei, "Boosting image captioning with attributes," *CoRR*, vol. abs/1611.01646, 2016.

[21] A. P. Mathews, L. Xie, and X. He, "Semstyle: Learning to generate stylised image captions using unaligned text," *CoRR*, vol. abs/1805.07030, 2018.

[22] C. Gan, Z. Gan, X. He, J. Gao, and L. Deng, "Stylenet: Generating attractive visual captions with styles," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[23] X. Liu, H. Li, J. Shao, D. Chen, and X. Wang, "Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data," *CoRR*, vol. abs/1803.08314, 2018.

[24] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Optimization of image description metrics using policy gradient methods," *CoRR*, vol. abs/1612.00370, 2016.

[25] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge," *CoRR*, vol. abs/1609.06647, 2016.

[26] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[27] Y. Li, C. Huang, C. C. Loy, and X. Tang, "Human attribute recognition by deep hierarchical contexts," in *European Conference on Computer Vision*, 2016.

[28] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan, "Style transfer in text: Exploration and evaluation," *CoRR*, vol. abs/1711.06861, 2017.

[29] K. Carlson, A. Riddell, and D. N. Rockmore, "Zero-shot style transfer in text using recurrent neural networks," *CoRR*, vol. abs/1711.04731, 2017.

[30] U. Nagarajan, G. Kantor, and R. Hollis, "The ballbot: An omnidirectional balancing mobile robot," *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 917–930, 2014.

High Precision In-Pipe Robot Localization with Reciprocal Sensor Fusion

Dapeng Eagle Zhao[1] and William Whittaker[1]

***Abstract*— Manual measurement of U-235 deposits in uranium enrichment piping is a costly, time consuming, and labor-intensive process that is a well-known cost and schedule driver. Autonomous, robotic innovation enabled by this research is revolutionizing the measurement speed, quality and safety of this important operation. The huge advantage is the robot's ability to measure from inside the pipes. The upside is sensing the geometry, appearance and radiometry directly. The downside is the inability to know precise, absolute position of the measurements in very long pipe runs. This paper develops the unprecedented localization required for this purpose.**

This paper presents the precise localization method designed for in-pipe radiation measurement robots. Unlike other in-pipe localization methods, this approach achieves millimeter-level accuracy in hundred-foot runs and does everything on one robot without relying on tether cable or communicating devices out of pipe, which is a important feature for nuclear in-pipe robot. It overcomes challenges usually encountered by in-pipe localization such as long travelling distance and operation time, narrow view angle, and featureless environment.

The robot is equipped with encoders embedded in tracks which can record travelled distance, but encoder odometry drifts unacceptably due to slip and non-linearity. The robot also incorporates a laser rangefinder measuring the absolute distance, but rangefinder in long pipes has many misleading false measurements that are difficult to filter. The innovation here is to use one to filter/calibrate the other so that data used from the two sensors improve each other iteratively and reciprocally. The survey measurements accuracy of this localization method is proven by experiments comparing to ground truth and "zippering error" experiments that compare measurements to fixed features in pipes.

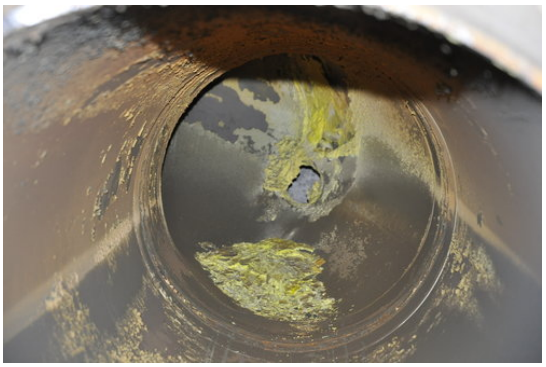


Fig. 1: Uranium holdup deposit in process piping

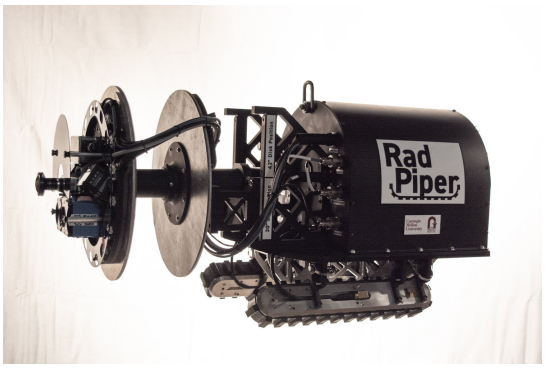


Fig. 2: U-235 deposit measurement pipe-crawling robot RadPiper

I. Introduction

A. Background

Vast amounts of U-235 remain in miles of piping that once enriched America's uranium. These immense facilities are now defunct and decommissioning is underway. An immense driver of decommissioning schedule and budget is a requirement to determine the exact grams of U-235 in every foot of that pipe before demolition.

To date, human workers in protective clothing have manually deployed detectors from the outside of these pipes to observe radiation emanating from the U-235 inside the pipes. This incurs operational disadvantages of clearing around pipes for access, hazards of elevated work, rad exposure, and manual data transcription. The technical disadvantages include faint signal from attenuation through pipe walls, inability to directly view the deposits, and inability to position a detector on the pipe's center-line.

Since decommissioning involves cutting pipes, the unique opportunity is to robotically deploy detectors from the inside rather than the outside to measure per-foot quantities of U-235. Compared to the current manual method, in-pipe robotic measurement achieves superior speed, accuracy and certainty. It does not requiring demolition for clear access around pipes. It provides video and geometric record of deposit acquired from inside pipes and precludes significant elevated human work.

An autonomous U-235 deposit measurement pipe-crawling robot is being developed by Carnegie Mellon University called RadPiper [1][2] (Fig. 2). RadPiper is a battery-powered and tetherless robot which self-steers using two tracks. A detector assembly is mounted on the front to acquire radiometric data. The robot is recovered from the same pipe opening from which it is launched, hence it drives the same distance out and back, measuring the same deposits twice. This achieves redundant radiometric and odometric measurements which adds further to statistical significance.

*Funding for this development was provided by the Office of Environment, Department of Energy of the US.

[1]Field Robotics Center, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, US dapengz at andrew.cmu.edu

For this in-pipe radiation measuring robot, localization is essential, because it is required to report the precise location of each radiation deposit measured. Also, since the robot will measure the same deposit twice running forward and backward, the locations of the two measurements must match up with each other precisely.

B. Existing solutions and Challenges

Several prior approaches have realized robotic in-pipe localization.

Encoder odometry is a simple, effective and well-studied robot localization method. However, localization entirely relying on encoder odometry is vastly insufficient for the required precision and certainty. Encoding suffers from accumulated error over long travelling distance. Odometry also measures the steered path which is always longer than the straight-line pipe distance. These problems cannot be completely modelled or predicted.

Tether cable is adapted by some in-pipe robots for purposes of energy, communication, safety and recovery, as well as odometry. Martra and Tur [3] measured the length of a tether cable to acquire traveled distance information. However, this approach is not adaptable for some in-pipe robots which are not equipped with tether for varieties of reasons. For example, as an fully-autonomous robot, RadPiper precludes tether for reasons of contamination and handling.

Some localization exploits ELF-EP communication between an in-pipe robot and sensor station set up out of the pipes [4][5]. However, in many cases such as nuclear pipes for RadPiper, it is difficult or impossible to set up sensor stations or other localization assisting equipment along a pipe.

Visual odometry is an approach for odometry of some in-pipe robots [6][7]. It is not typically very robust because its performance is highly affected by how many available valid features can be found on an inner pipe surface. In some cases such as uranium enrichment piping, only a few recognizable features can be found sparsely (Fig. 3).

The precision of the prior localization methods is also an issue. With ELF-EP communication method [4][5], from the experiments conducted by Qi et.al, the error is about 0.75ft (0.23m) to 1.5ft (0.45m). In the research work done by Hansen et.al with visual odometry [6][7], the error was found as 0.84ft (256.4mm) when the travelling distance is about 100ft (30483mm). However, radiation measurement robots require much higher precision.

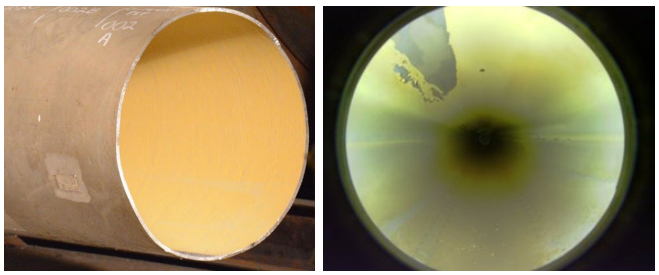


Fig. 3: No feature suitable for Visual Odometry(left); Occasional visual features are too rare and indistinct for Visual Odometry(right)

C. Technical approach

The localization method developed here is based on data fusion of two sensors: laser rangefinder and track encoders.

The rest of this paper will present how reciprocal and iterative sensor data fusion is done in details and its corresponding testing result from RadPiper.

II. Reciprocal sensor fusion odometry

The robot RadPiper is driven by a pair of trucks. There is an encoder embedded in each track which can record travelled distance. RadPiper also incorporates a laser rangefinder which measures the absolute distance. The rangefinder is intended to measure the distance between the robot and the reflecting surface at the pipe end. The whole setup is shown in Fig. 4.

In this section, the paper will respectively discuss about problems of raw data from each sensor in details and present how the problems of one sensor can be improved by the other one iteratively and reciprocally. With the improved data, the paper will then discuss how data fusion is done to estimate the robot's trajectory. The general overall work flow of this localization method is shown in Fig. 5.

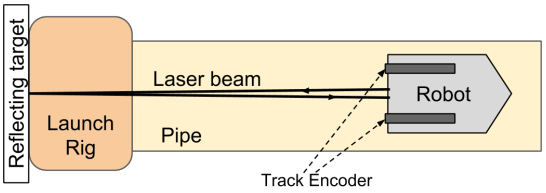


Fig. 4: Laser ranging and track encoding for localization

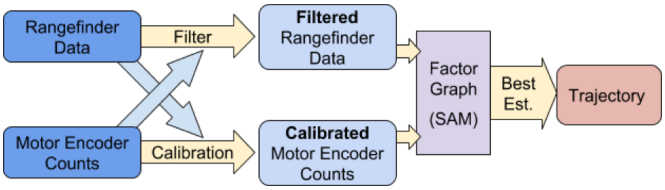


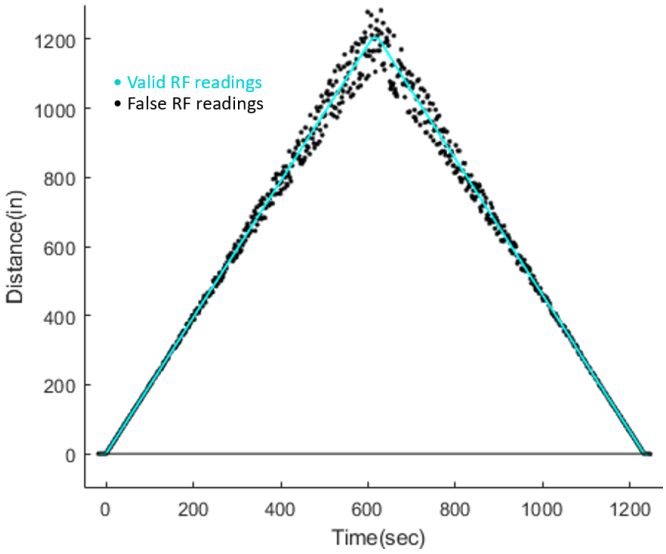
Fig. 5: Work flow for high precision in-pipe Localization with Reciprocal Sensor Fusion

A. Rangefinder Data Filter

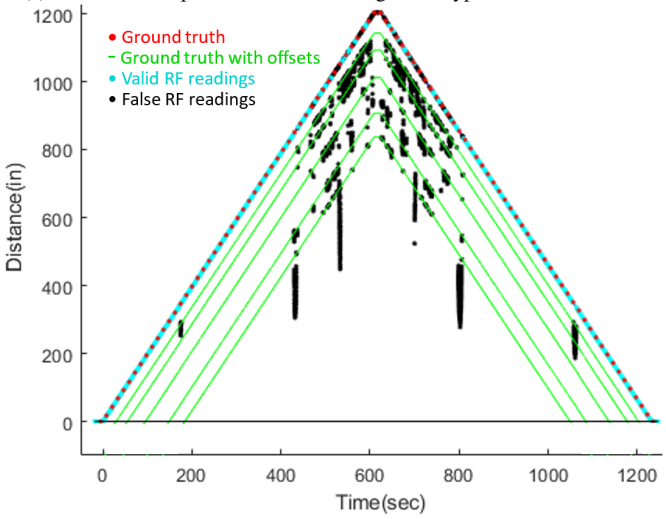
A laser rangefinder is installed on RadPiper to measure the distance back to the end of a pipe. However, typically rangefinders do not perform well in pipes, because in pipe rangefinders take many false readings. The false readings are not just valid reading with random noise (Fig. 6a), but have certain patterns that is challenging for traditional filter method to deal with.

Real rangefinder data from one test run is plotted in Fig. 6b. The horizontal axis is time in second, and the vertical

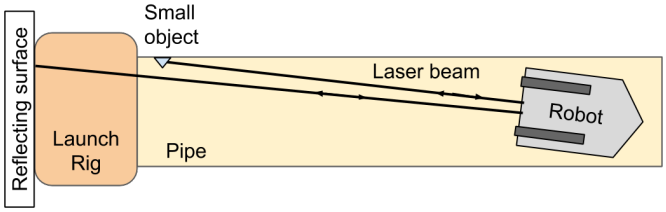
axis is distance in foot. The V-shape red line is the actual distance between the robot and the end of pipe. It increases and then decreases, indicating the robot drives into the pipe and traverse back to the starting point.



(a) Distance-time plot of laser Rf readings with typical Gaussian noise



(b) Distance-time plot of laser Rf readings acquired while driving out and back in 1210-inch pipe during 1235 seconds.



(c) Rangefinder false reading

Fig. 6: Laser Rangefinder Illustrations

The aqua and black dots on the plot are measurements from the rangefinder. Aqua data points are valid readings which lie right on the red line. Black data points are false readings which do not reflect the real travelling distance of the robot.

The false readings are not entirely random. Most of them can be found on the green lines, each of which is the correct trajectory being shifted by a certain distance. In fact, false measurements are caused by laser beam reflecting off pipe walls or other objects in pipe instead of the intended surface behind launch rig, as demonstrated in Fig. 6c. The small object on the diagram can be a deposit lump or convex part of unsmooth pipe interior surface. False readings measure the distance from the robot to this kind of objects, which are always shorter than the actual distance, so in Fig. 6a almost all black false data points fall below the red line.

The measurements are almost all aqua at shorter distance within 70ft. As the distance increases, less and less aqua readings but more and more black false ones appear. It is because that at long distance the rangefinder's view angle is narrowed due to the nature of the shape of pipes. Therefore, the chance of false readings happening increases as the robot drives deeper into the pipe.

Though at long distance valid data points in aqua become very few, whenever there are valid measurements it is always right on red line reflecting the distance precisely.

Overall, laser rangefinder is a good sensor for localization because it is highly precise if the measurement is valid. However, at long distance valid measurements becoming very sparse and intermittent. Furthermore, false readings from rangefinder at long distance are not any typical noise. Other than offsetting, they are very similar to the correct readings, which makes it very difficult to filter.

To filter out the false readings, different approaches were proposed. If a line can be found that is close enough to the trajectory, a region with a carefully chosen width then can be set around this line with boundaries above and below. Readings out of this region will be considered false readings to be eliminated. In order to find a good estimating line, line fitting methods like RANSAC[8] or votting scheme Hough transform[9] were adapted, but it did not perform well because the robot steers forward, its travelling path is not straight resulting the straight-line pipe distance does not increase linearly. Therefore, it is nearly impossible find a line that can work well for filtering.

One of the reasons why methods above would fail is that they try to solve the problem with rangefinder data standalone. To filter well, it is necessary to have a good estimate of the real trajectory, while the unfiltered raw rangefinder data including both correct and false readings form multiple lines of trajectory ambiguously.

In the method that is being presented, encoder odometry is used for this purpose being a trajectory estimate to filter rangefinder data. Though encoder odometry has drifting issue over long distances, in this method, this issue is avoided by iteration, which allows us to only look at a small distance for encoder odometry at each time.

After syncing different sensors' measurements, encoder counts and rangefinder readings share the same time stamps of measurements. Rangefinder readings are noted as $Rf[t]$ and encoder counts are noted as $Ec[t]$, while $t \in T$, T is a set of discrete time stamps. During filtering, $Loc_{est}[t]$ as

estimated location will also be generated at the same time as a reference value to help identify false rangefinder reading. The strategy of filtering rangefinder reading with encoder counts is presented in a iterative manner as follows:

- For $t = 0$, $Loc_{est}[0] \leftarrow 0$.
- For $t = k(k \geq 1, k \in \mathbb{N})$, $Loc_{est}[k+1]$ is obtained:
  1) Estimate $Loc_{est}[k]$ with encoder counts:
  $$Loc_{est}[k+1] = Loc_{est}[k] + \frac{Ec[k] - Ec[k-1]}{C}$$
  Here C is a coefficient that converts encoder counts to actual distance and has unit as "counts/inch". Its value is calculated from track and motor configurations like gear ratio, track diameter, etc.
  2) Now with a reasonable estimated location value, the actual rangefinder reading at the moment t=k will then be examined. If the difference between the actual rangefinder reading and the estimated location is larger than a selected threshold, this rangefinder reading will be marked as false reading and eliminated. If the difference is within the threshold, which means the rangefinder reading is valid and reliable, $Loc_{est}[k]$ will then be updated with this rangefinder measurement. The threshold, $thres$, is an experimental value affected by the clearance between valid and false readings and the requirement on the final accuracy.
  if $|(Loc_{est}[k] - Rf[k])| > thres$
  then $Eliminate\ this\ reading$
  else $Loc_{est}[k] \leftarrow Rf[k]$

With this approach, the false rangefinder readings can be precisely marked and eliminated. After well-filtering, all rangefinder data are valid and reliable now.

B. Track Encoder Counts

The robot has two tracks on both left and right side of the chassis, and each track is equipped with a encoder. The average of the two encoder counts reflects the travelling distance of the robot center proportionally:

$$Ec[t] = \frac{Ec_{Left}[t] + Ec_{Right}[t]}{2}$$

, where $Ec_{Left}[t]$ is the encoder counts from left track and $Ec_{Right}[t]$ is from the right track.

$$Ec[t] \propto Loc[t]$$

, where $Loc[t]$ is the robot's travelling distance, also the location when taking start point as 0.

As discussed earlier, encoder odometry is not reliable over a long distance due to drifting issues. Drifting may be caused by slippage, robot active steering, robot climbing over lumps or other accidental situations. However, after filtering rangefinder data, it becomes possible to calibrate encoder odometry using valid rangefinder reading as landmarks.

Encoder odometry is forced to meet its corresponding rangefinder reading periodically to prevent error being accumulating. A step distance is set, so that whenever location is changed, increased or decreased, by the step distance, the encoder odometry is multiplied with a coefficient to equal to the rangefinder reading taken at the same time. If the rangefinder reading was marked as false reading at the current time stamp, the next encoder odometry data point will then be considered instead in the same way. The detailed implementation is presented below. Only the first half of the trajectory, the part of robot driving out, is presented here.

- Initially, $Ec[0]$ is considered as "calibrated". Therefore, the flag j for "the last calibrated encoder reading" is set as 0. Searching index k is set as 0.
- Loop this section until the end of the first half of the trajectory:
  1) Keep increasing index k,
  until $Rf[k] - Rf[j] > Dist_{step}$,
  $Dist_{step}$ is the step distance
  2) $Ec[j:k] \leftarrow Ec[j:k] * \frac{Rf(k) - Ec(k)}{Rf(k) - Rf(j)}$
  3) $j \leftarrow k$

In practice, the step distance is set as 3 foot.

After the operation above, encoder odometry's drifting problems, if there are, are restricted within each 3-foot section. Up to now, encoder odometry is well-calibrated by filtered rangefinder data, and ready for the following steps.

In reality the two process, rangefinder data filtering and encoder odometry calibration, happen concurrently.

C. Information Fusion - Generating Trajectory

With all the data from rangefinder and encoder, a factor graph can be built for state estimation to generate the robots trajectory. In this process, toolbox GTSAM [10] was adpated. In the factor graph, position estimates are inserted as nodes and measurements as edges. Edges for encoder odometry were inserted with larger variances and edges for rangefinder data with smaller variances, because rangefinder gives absolute and accurate measurement whenever it is available. A trajectory will be formed in the end by optimizing each state in the graph to reach the maximum likelihood.

III. Testing Result

In order to evaluate this localization method, two tests were conducted wit seven 100ft-long test runs.

A. Test I: Ground Truth Comparison

Result from reciprocal sensor fusion odometry is compared with the ground truth which is measured by a total station. A total station is an electronic optical instrument usually used for surveying and building construction and it measures the distance through a modulated infrared signal emitted by itself and reflected by a prism installed on the object[11]. In our case, a prism is installed on the robot for total station to measure and offer ground truth. The whole setup for testing is shown in Fig. 7.

Error in this test is considered as the absolute value of the difference between reciprocal sensor fusion localization

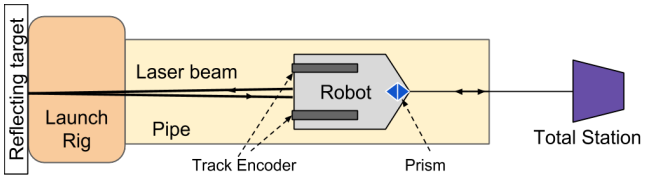


Fig. 7: Total station setup for acquiring ground truth

result and ground truth measured by total station.

$$E_1(t) = |Loc(t) - Gt(t)|$$

, $E_1(t)$ is the error, $Loc(t)$ is the localization result, and $Gt(t)$ is the ground truth. Error $E(t)$ from one random-selected test run is plotted in Fig. 8. The Error-Time plot shows that error is below 0.3 inch almost at almost all time, and only very few extreme error values reached over 0.5 inch. The statistical analysis of the ground truth comparison error from seven test runs is shown in Table I.

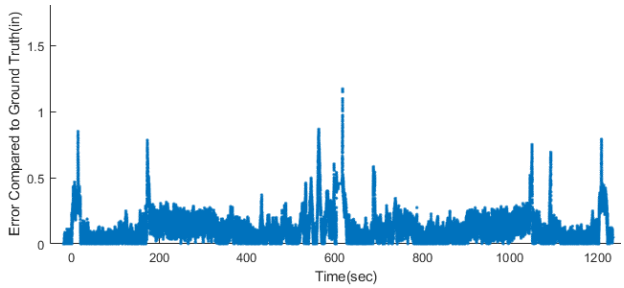


Fig. 8: Error of Reciprocal Sensor Funsion Localization vs. Ground Truth from one Test Run

TABLE I: Error (inch) of Reciprocal Sensor Fusion Localization vs. Ground Truth

Test Run	Max(E_1)	Mean(E_1)	Var(E_1)	Std(E_1)
1	0.96	0.10	0.0107	0.10
2	1.07	0.11	0.0155	0.12
3	0.99	0.12	0.0245	0.16
4	0.94	0.12	0.0125	0.11
5	1.17	0.11	0.0116	0.11
6	1.14	0.14	0.0259	0.16
7	1.59	0.13	0.0276	0.17
Max.	1.59	0.14	0.0276	0.17
Ave.	1.12	0.12	0.0183	0.13

B. Test II: Block Test

As discussed earlier, during operation a uranium deposit will be detected twice on the robot's forward and backward running and it is very important that the two measured locations match up with each other precisely.

Therefore, a test was setup as shown in Fig. 9. In a 100-feet-long pipe, 25 wooden blocks were placed 4 feet apart from each other as "simulated deposits". The robot is deployed to run through the pipe and come back to the launch rig. Time stamps are recorded at the same time when blocks are detected by the Lidar installed on the robot. One block has two corresponding time stamps because the robot detects it twice on both forward and backward running. After post-processing, based on the blocks' time stamps, the corresponding locations can be found from the generated trajectory. Error between the two measured locations is called "zippering error" in this project.

$$E_2(n) = Block(n).Loc(t_f(n)) - Block(n).Loc(t_{b(n)})$$

, where n is the block number, $Block(n)$ indicates the n-th block, $E_2(n)$ is the "zippering error" of the n-th block, $t_f(n)$ is the time stamp recorded when the robot detects this block during forward running and so is $t_b(N)$ for backward, and $Loc(t)$ is the reciprocal sensor fusion localization result. This "zippering error" from one randomly selected test run is shown in Fig. 10, and the overall statics of all seven test runs is shown in Table.2.

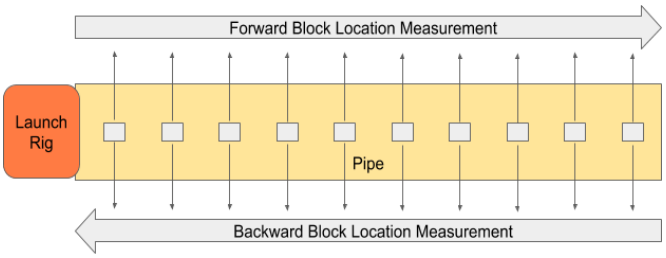


Fig. 9: Block test setup illustration

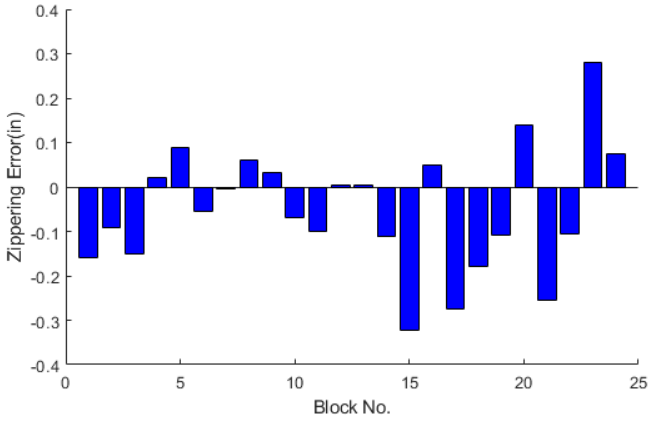


Fig. 10: "Zippering Error" from One Test Run

TABLE II: "Zippering Error" Statistics Overview

Test Run	Max(E_1)	Mean(E_1)	Var(E_1)	Std(E_1)
1	0.41	0.13	0.0106	0.10
2	0.46	0.14	0.0145	0.12
3	0.57	0.13	0.0178	0.13
4	0.20	0.08	0.0023	0.05
5	0.32	0.12	0.0087	0.09
6	0.36	0.11	0.0100	0.10
7	0.43	0.13	0.0175	0.13
Max.	0.57	0.14	0.0178	0.13
Ave.	0.39	0.12	0.1163	0.10

IV. Conclusion

Reciprocal sensor fusion odometry achieves precise and certain in-pipe localization unachievable by previous method. This is particularly for long runs where encoder drift is significant and absolute range sensing is intermittent.

The error between reciprocal sensor fusion localization and ground truth measurements in large diameter hundred-foot piping is typically around 0.1 inch.

The RadPiper robot localizes any feature during its forward and backward traverses. Hence, a metric of precision is the extend to which its forward and backward localization of a given feature are identical. For 24 objects over a distance of nearly 100 feet, the maximum of the zippering error is 0.4 inch, the mean error is 0.11 inch and the standard deviation of the error is 0.09 inch.

ACKNOWLEDGMENT

The author would like to thank Dr. William Red Whittaker, as well as the RadPiper team, for the guidance and support.

Also, thanks to Hong Kong Polytechnic University for sponsoring Dapeng Zhao to participate in this research activity.

Last but not least, special thanks to Rachel Burcin, John Dolan, as well as the rest of RISS team.

References

[1] H. Jones, W. Whittaker, O. Sapunkov, T. Wilson, D. Kohanbash, S. Maley, J. Teza, E. Fang, M. McHugh, I. Holst, C. Ng, and R. Riddle, "Robotic NDA of holdup deposits in gaseous diffusion piping by gamma assay of uranium-235," in *WM2018 Conference*, Phoenix, Arizona, USA, Mar. 2018.

[2] H. Jones, S. Maley, T. Wilson, R. Riddle, M. Reibold, W. Whittaker, D. Kohanbash, L. Papincak, and W. Whittaker, "Robotic nda of holdup deposits in gaseous diffusion piping by gamma assay of uranium-235," in *WM2018 Conference*, Phoenix, Arizona, USA, Mar. 2018.

[3] A. C. Murtra and J. M. M. Tur, "IMU and cable encoder data fusion for in-pipe mobile robot localization," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, apr 2013. [Online]. Available: https://doi.org/10.1109/tepra.2013.6556377

[4] H. Qi, X. Zhang, H. Chen, and J. Ye, "Tracing and localization system for pipeline robot," *Mechatronics*, vol. 19, no. 1, pp. 76–84, feb 2009. [Online]. Available: https://doi.org/10.1016/j.mechatronics.2008.06.001

[5] H. Chen, H. Qi, and X. Zhang, "Research of pipeline robot tracing & localization technology based on ELF-EP communication," in *2008 IEEE International Symposium on Industrial Electronics*. IEEE, jun 2008. [Online]. Available: https://doi.org/10.1109/isie.2008.4676936

[6] P. Hansen, H. Alismail, P. Rander, and B. Browning, "Monocular visual odometry for robot localization in LNG pipes," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, may 2011. [Online]. Available: https://doi.org/10.1109/icra.2011.5979681

[7] P. Hansen, H. Alismail, B. Browning, and P. Rander, "Stereo visual odometry for pipe mapping," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2011. [Online]. Available: https://doi.org/10.1109/iros.2011.6094911

[8] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, jun 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[9] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[10] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[11] B. F. Kavanagh and S. J. G. Bird, *Surveying principles and applications*, 4th ed. New Jersey: Prentice Hall, 1996.

Extrinsic Calibration Algorithm between a Stereo Visual System and a 3D LiDAR

Yang Zhou[1]

Abstract— **Camera and light detection and ranging (LiDAR) are frequently used for perception in real-world applications. The combination of these heterogeneous sensors bring advantages of both kinds of sensors and get accurate and promising results. The extrinsic calibration of camera and LiDAR system is a prerequisite for robot perception applications, the robustness and usability are required in research and industry. Previous works have provided several extrinsic calibration methods including target-based works and target-less works. Although there are many works dealing with calibration of a monocular camera and a 3D LiDAR, the solutions of the calibration of the stereo visual system and 3D LiDAR still have not meet the requirement of the real-world application yet. In this work, we designed an extrinsic calibration algorithm which can be used for calibration of a stereo visual system and a 3D LiDAR system. Our method can estimate extrinsic parameters between a stereo visual system and a 3D LiDAR with only one checkerboard pose. We can further improve the result by a joint non-linear optimization considering LiDAR disparity constraint using multiple poses.**

I. Introduction

Nowadays, the number of different sensors mounted on robots is increasing. Different sensors have different characteristics in real-world robotics applications which enable robots to perceive the environment in challenging situations. To utilize the advantages of different sensors, the fusion system of different modalities have been widely applied to perception, navigation, and mapping applications. The transformation relationship between different sensors is required for sensor fusion system to align information in a common coordinate system. Since robot perception applications are highly relying on the intrinsic parameters and extrinsic parameters of different sensors, the calibration of multiple heterogeneous sensors needs to be precise. The calibration method needs to be robust for different settings and needs to be convenient and user-friendly.

LiDAR and stereo visual system have different characteristics. Light detection and ranging (LiDAR) is a range sensor can obtain accurate range measurements using laser beams. 3D LiDAR uses multiple laser beams to produce precise 3D point cloud, so it can sense the geometry feature of the environment. Visual Camera can obtain color and texture information of the scene. The stereo visual system has two cameras which can capture 3D information of scene based on feature matching between two cameras. To utilize the advantages of the stereo visual system and the 3D LiDAR, we need to calibrate these two different sensors.

The research focusing on this problem, however, is limited. Although there are some works focusing on the extrinsic calibration of 2D LiDAR and stereo visual system and some works focusing on the calibration of 3D LiDAR and monocular camera, there are very few works dealing with extrinsic calibration between a stereo visual system and 3D LiDAR. Among these works, some methods [1], [2], [3], [4] exploit mutual information between different sensors. Some methods [5], [6], [7] use special designed calibration target. And some works [8], [9], [10], [11] use a checkerboard which is common to users as the calibration target. We also choose checkerboard as our calibration target because it enables us to calibrate the intrinsic and extrinsic parameters of the stereo visual system and LiDAR simultaneously.

To tackle the extrinsic calibration problem, we need to find constraints between the stereo visual system and 3D LiDAR to find the geometry relationship between these two different sensors. The plane of checkerboard has been used as geometry constrained in the extrinsic calibration of a 2D LiDAR and a camera. They calculate the extrinsic parameters only by plane correspondences which require 3 poses as a minimum number. Line correspondences and point correspondences are also explored in many works on the extrinsic calibration of a 2D LiDAR and a camera. To reduce the minimum number requirement of poses, we obtain 3D line and plane correspondences according to 4 boundaries of checkerboard and 1 plane of the checkerboard. We can estimate 2D line and 3D plane feature on the image based on the line detection algorithm and intrinsic parameters of the camera. 3D line feature on the image is calculated by the intersection of the back-projected plane of 2D line extracted from the image and the 3D plane of the checkerboard extracted from the feature points on the checkerboard and the intrinsic parameters of the camera. We can estimate 3D line and 3D plane feature based on LiDAR point cloud using the RANSAC[12] algorithm to eliminate outliers. Using 1 plane correspondence and 2 of 4 line correspondences, we can estimate extrinsic parameters of a monocular camera and a 3D LiDAR with one pose using a close-formed solution. After calibrating each single cameras with 3D LiDAR, we globally optimize extrinsic parameters with LiDAR disparity constraint by non-linear optimization.

The contribution of this paper can be summarized as:

1) Extend extrinsic calibration of a monocular camera and a 3D LiDAR [13]to extrinsic calibration of a stereo visual system and a 3D LiDAR.

[1]Yang Zhou is with School of Information Science and Technology, ShanghaiTech University, Shanghai, China. This work is done during Robotics Institute Summer Scholar program in the Robotics Institute, Carnegie Mellon University. zhouyang@shanghaitech.edu.cn

2) We reduce the minimum requirement of poses from 3 to 1 by utilizing 3D correspondences of lines and planes. The system can be extended to extrinsic calibration of multi-camera and multi-LiDAR system.
3) We introduce LiDAR disparity constraint to improve the result with multiple poses by non-linear optimization.
4) We develope an extrinsic calibration software in C++ with high efficiency and robustness.

II. Related Works

There are different methods purposed to solve multiple sensors calibration problem, here we will talk about related works according to different problem categories about the camera to LiDAR calibration.

For the monocular camera to LiDAR calibration problem, there are different types of approaches based on different correspondences. Some methods do not rely on the calibration target. In [14] and [4], they make use of mutual information between LiDAR reflectivity and camera intensity to do the extrinsic calibration outdoors without a calibration target. In [15] relies on PnP algorithm [16] using manually selected points correspondences. In [17], the deep neural network is applied on extrinsic calibration to solve this problem by end-to-end training.

Some methods use a rectangle calibration target such as a checkerboard. [8] uses a checkerboard to do extrinsic calibration between a perspective camera and a 2D LiDAR, they use the plane-line correspondence established by LiDAR points on plane and plane parameters estimated in the camera coordinate system, their algorithm requires at least 5 poses to get the extrinsic parameters. [9] use plane-plane correspondence established by estimating plane parameters from 3D LiDAR and camera to estimate initial rotation matrix and translation vector, then they use non-linear optimization to refine the result by minimizing point to plane distance. Their method needs at least 3 poses. Instead of using one checkerboard, [10] uses several checkerboards in front of sensors to do the extrinsic calibration with one pose, which avoids moving checkerboards. Methods based on plane constraint can be easily extended to the multi-sensors system, [11] calibrate the extrinsic parameters between an omnidirectional camera and a 3D LiDAR. When estimating plane parameters from the image and 3D LiDAR, a plane from a farther distance cannot be estimated as precisely as the plane from a closer distance. [13] can do extrinsic calibration between a camera and 3D LiDAR using lines and planes correspondences in one pose.

Some methods rely on specifically designed calibration target. In [5], arbitrary trihedron is used to estimate the extrinsic parameters with 2 poses. [6] uses a discontiguous calibration target to emphasize 2D LiDAR information to estimate extrinsic parameters between a camera and a 2D LiDAR. In [7], the v-shaped target is used to utilize the boundary of the target captured by LiDAR. These works use the target with a specially designed pattern to exploit the boundary information. The boundary information can be detected by LiDAR and visual system as correspondences, which can reduce the number of poses required for extrinsic calibration.

For the stereo camera to LiDAR calibration problem, [18] calibrate a 3D LiDAR and a stereo visual system using inertial measurement unit (IMU). With the help of inertial data, they used a bright spot as the only calibration target. Their framework can be extended to the multi-camera network. In [19], a board with circular hole pattern is used to calibrate a 3D LiDAR and a stereo visual system. The circular hole pattern can be detected in camera frame and LiDAR frame robustly. In [20], the paper presents an extrinsic calibration algorithm between a stereo vision system and a 2D LiDAR-based on the 3D reconstruction of the checkerboard. They use 3D corner points of checkerboard obtained by stereo camera system to do triangulation, solve least-square estimation of the 3D plane of checkerboard and use non-linear optimization to optimize extrinsic parameters. [21] used the particle swarm optimization algorithm (PSO) to estimate extrinsic parameters and fuse information of a stereo camera and a LiDAR without the aid of another calibration target. [22] proposed a method to calibrate a 2D LiDAR and a multi-camera system, it decouples the problem into two hierarchical level optimization problem without using any calibration target.

Comparing to the above works, our method utilizes 3D planes and lines correspondences between camera and LiDAR, we use checkerboard to extract the plane and boundary of checkerboard both from camera and LiDAR. Considering there will be some cases that intrinsic parameters of the camera are unknown, the checkerboard can be used to estimate intrinsic parameters at the same time. The boundary of the checkerboard can be extracted easily because of the visual feature and reflectance feature of the checkerboard. To consider the stereo visual system and 3D LiDAR globally, our method refines the initial optimization result according to the geometry constraint of the stereo visual system which leads to the better result.

III. Methods

A. Problem Formulation

This section will describe the formulation of the extrinsic calibration problem of a stereo visual system and a 3D LiDAR. The extrinsic calibration of a stereo visual system and a 3D LiDAR is to estimate the relative rotation and translation between two cameras and the 3D LiDAR. The relationship of the three sensors is shown in Fig. 1. We use $\mathbb{R}^3_{C_1}$ as the coordinate system of the left camera, $\mathbb{R}^3_{C_2}$ as the coordinate system of the right camera, and $\mathbb{R}^3_l$ as the coordinate system of the 3D LiDAR.

For camera system, we will do intrinsic calibration to both cameras to get the intrinsic parameters of cameras and undistort image by estimated distortion coefficients by the method described in [8] Then we can model both cameras by pinhole camera model. We denote $\mathbf{K}^{C_1}$ as the intrinsic matrix of the left camera and $\mathbf{K}^{C_2}$ as the intrinsic matrix of the right camera. The points in left camera coordinate

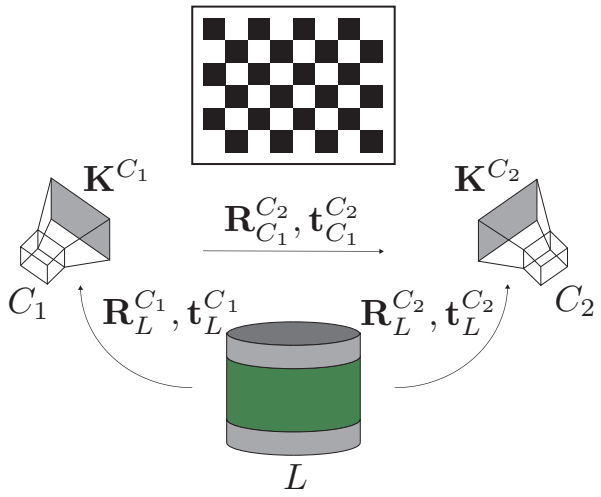


Fig. 1. Sensors Relashionship

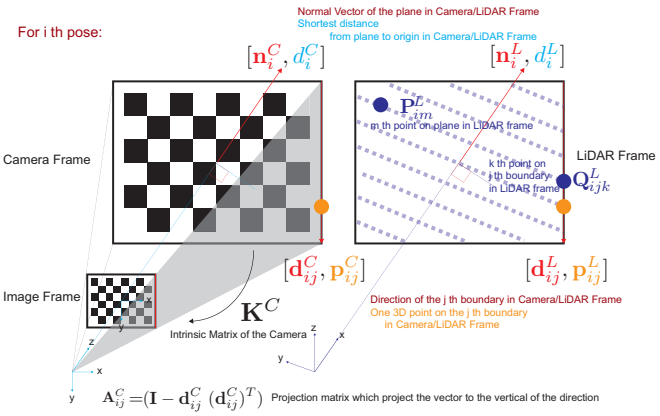


Fig. 2. Notation of plane and line parameters

system are presented as $\mathbf{P}^{C_1} : (X^{C_1}, Y^{C_1}, Z^{C_1}) \in \mathbb{R}^3_{C_1}$, and points in right camera coordinate system are presented as $\mathbf{P}^{C_2} : (X^{C_2}, Y^{C_2}, Z^{C_2}) \in \mathbb{R}^3_{C_2}$. The pixels of points in left camera image can be described as (x^{C_1}, y^{C_1}). The pixels of points in right camera image can be described as (x^{C_2}, y^{C_2}). The relationship between the camera coordinate system and camera image frame can be described as below.

$$\mathbf{K}^{C_1}\mathbf{P}^{C_1} = \begin{bmatrix} x^{C_1} \\ y^{C_1} \\ 1 \end{bmatrix} Z^{C_1} \tag{1}$$

$$\mathbf{K}^{C_2}\mathbf{P}^{C_2} = \begin{bmatrix} x^{C_2} \\ y^{C_2} \\ 1 \end{bmatrix} Z^{C_2} \tag{2}$$

For LiDAR system, we convert the distance from the direction of each scan points in polar coordinates obtained by 3D LiDAR to the form of Cartesian coordinates as $\mathbf{P}^L \in \mathbb{R}^3_l$.

For extrinsic parameters, we decouple the transformation relationship between cameras and 3D LiDAR as rotation matrix $\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2} \in SO(3)$ and $\mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2} \in \mathbb{R}^3$. Let $(\mathbf{R}_L^{C_1}, \mathbf{t}_L^{C_1})$ and $(\mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_2})$ be the relative rotation matrix and translation vector of left camera to 3D LiDAR and right camera to 3D LiDAR. Let $(\mathbf{R}_{C_1}^{C_2}, \mathbf{t}_{C_1}^{C_2})$ be the relative rotation matrix and translation vector from the left camera to the right camera. The relationship of three sensors can be described as below:

$$\mathbf{P}^{C_1} = \mathbf{R}_L^{C_1}\mathbf{P}^L + \mathbf{t}_L^{C_1} \tag{3}$$

$$\mathbf{P}^{C_2} = \mathbf{R}_L^{C_2}\mathbf{P}^L + \mathbf{t}_L^{C_2} \tag{4}$$

$$\mathbf{P}^{C_2} = \mathbf{R}_{C_1}^{C_2}\mathbf{P}^{C_1} + \mathbf{t}_{C_1}^{C_2} \tag{5}$$

For ith pose, we extract 3D plane features from both cameras and 3D LiDAR. The 3D plane parameters in cameras are described as $[\mathbf{n}_i^{C_1}; d_i^{C_1}]$ and $[\mathbf{n}_i^{C_2}; d_i^{C_2}]$. $\mathbf{n}_i^{C_1}, \mathbf{n}_i^{C_2}$ represent the normal vector of the plane in the camera frame, $d_i^{C_1}, d_i^{C_2}$ represent the distance from the plane to the origin of the camera coordinate system. The 3D plane parameters in LiDAR are described as $[\mathbf{n}_i^L; d_i^L]$. $\mathbf{n}_i^L$ represents the normal vector of the plane in the LiDAR frame, d_i^L represents the distance from the plane to the origin of the LiDAR coordinate system. For any point $(X_1, Y_1, Z_1) \in \mathbb{R}^3_{C_1}$ that lie on plane i in left camera , any point $(X_2, Y_2, Z_2) \in \mathbb{R}^3_{C_2}$ that lie on plane i in right camera, point $(X_l, Y_l, Z_l) \in \mathbb{R}^3_l$ that is the centroid point of plane i in 3D LiDAR,

$$[\mathbf{n}_i^{C_1}, d_i^{C_1}] \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ -1 \end{bmatrix} = \mathbf{0} \tag{6}$$

$$[\mathbf{n}_i^{C_2}, d_i^{C_2}] \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ -1 \end{bmatrix} = \mathbf{0} \tag{7}$$

$$[\mathbf{n}_i^l, d_i^l] \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ -1 \end{bmatrix} = \mathbf{0} \tag{8}$$

For the ith pose, we also extract 3D line features from both cameras and 3D LiDAR. The the jth 2D line in left camera and right camera are described as $l_{ij}^{C_1}$, $l_{ij}^{C_2}$ which contains image-space coordinates of starting points and ending points. The the jth 3D plane in left camera and right camera are denoted as $[\mathbf{d}_{ij}^{C_1}, \mathbf{p}_{ij}^{C_1}]$ and $[\mathbf{d}_{ij}^{C_2}, \mathbf{p}_{ij}^{C_2}]$, where $\mathbf{d}_{ij}^{C_1}, \mathbf{d}_{ij}^{C2} \in \mathbb{R}^3$ are derections of 3D lines in both cameras and $\mathbf{p}_{ij}^{C_1}, \mathbf{p}_{ij}^{C2} \in \mathbb{R}^3$ are points lie on 3D lines in both cameras. We use $\mathbf{A}_{ij}^{C_1}$ and $\mathbf{A}_{ij}^{C_2}$ to represent the projection matrix $(\mathbf{I} - \mathbf{d}_{ij}^{C_1}(\mathbf{d}_{ij}^{C_1})^T)$ and $(\mathbf{I} - \mathbf{d}_{ij}^{C_2}(\mathbf{d}_{ij}^{C_2})^T)$.

The jth 3D line in 3D LiDAR can be described as $[\mathbf{d}_{ij}^L, \mathbf{p}_{ij}^L]$, where $\mathbf{d}_{ij}^L$ is the direction of the 3D line and $\mathbf{p}_{ij}^L$ is the centroid of points lie on the 3D line. We denote the kth point lie on the jth 3D line in 3D liDAR as $\mathbf{Q}_{ijk}^L$, and the mth point lie on the jth 3D plane in 3D LiDAR as $\mathbf{P}_{im}^L$. In ith pose, the number of points on the plane is denoted as N_i, the number of points on the jth boundary is denoted as K_{ij}.

In this paper, $L_{ij}^{C_1}$, $L_{ij}^{C_2}$ and L_{ij}^{L} represent the jth line in the ith pose of two cameras and 3D LiDAR, $\pi_i^{C_1}$, $\pi_i^{C_2}$ and π_i^{L} represent plane in the ith pose. The plane and line parameters are also illustrated in Fig. 2. We use $\cdot$ to represent the dot product.

B. System Description

We implement the extrinsic calibration algorithm of the stereo visual system and the 3D liDAR in C++. We will discuss the structure of the calibration software. The basic structure of the system is shown in [Fig].

To use this software, users need to use ROS [23] to record several bags containing images from both cameras and point clouds from 3D LiDAR. If intrinsic parameters of both cameras are not provided, users need to place the checkerboard in different positions to make sure that all image area is covered by different positions to get accurate intrinsic parameters. The point cloud and images will also be used to calibrate the extrinsic parameters. Users record images and point clouds for several seconds in rosbag at each position. Users need to provide a cuboid in LiDAR coordinate system which contains the checkerboard to help locate the checkerboard in the point cloud. Then this information are regarded as the input of the system.

Firstly, the system will use intrinsic calibration [8] built-in OpenCV [24] to calibrate the intrinsic parameters of both cameras using images containing checkerboard, then undistorted images will be stored in a folder for further processing.

In the second step, stereo calibration in OpenCV will be used to obtain initial extrinsic parameters of both cameras.

In the third step, 3D planes and lines feature from both cameras and 3D LiDAR are extracted. LSD method [8] is used to extract 2D lines from images, and plane parameters in cameras are estimated using checkerboard features. 3D planes in 3D liDAR are estimated by RANSAC [25], the 3D lines in 3D planes in LiDAR frame are also estimated by RANSAC after denoising.

In the fourth step, we use the SVD method to give out the initial guess of extrinsic parameters between each camera and the 3D liDAR.

In the fifth step, we utilize the LiDAR disparity constraint to refine the extrinsic calibration result globally by Levenberg-Marquardt algorithm[26].

C. Feature Extraction of Calibration Target

In this section, we will introduce the steps of feature extraction in detail.

1) Planes and Lines Extraction from Camera: We will consider feature extraction in cameras, without loss of generality, we will discuss it in the left camera.

The 3D plane parameters $[\mathbf{n}_i^{C_1}, d_i^{C_1}]$ can be computed by homography using checkerboard. After extracted all 2D lines by LSD[27] method, we can extract several line segments on the four boundaries of the checkerboard. After line fusion, we can get starting points and ending points $[(x_s, y_s), (x_e, y_e)]$ of a line. A sample result of 2D boundary detection is shown in Fig. 3. We can get back-projected 3D plane of the line which is the plane containing the 3D line and origin of $\mathbb{R}_{C_1}$ as $[K_{C_1} \cdot ([x_s, y_s, 1]^T \times [x_s, y_s, 1]^T); 0]$. Then we can get the 3D plane parameters by the intersection of the back-projected 3D plane and the 3D plane.

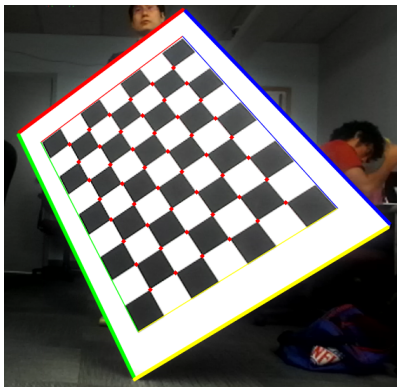


Fig. 3. Planes and Lines from Camera

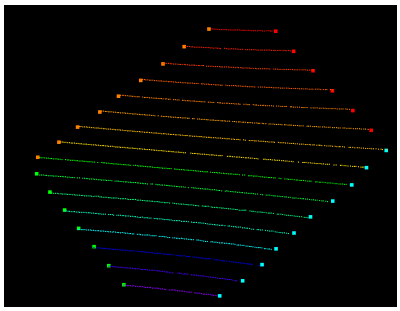


Fig. 4. Planes and Lines from 3D LiDAR

2) Planes and Lines Extraction from 3D LiDAR: For 3D LiDAR, we use user-provided cuboid to help locate the checkerboard in the point cloud. The 3D plane parameters can be computed by using the RANSAC algorithm to extract points on the checkerboard plane. Then we can filter points on scanlines according to the length of the consecutive segments to get a refined point cloud of the plane.

After getting the point cloud of the plane, we apply line fitting on each scanline to filter out outliers by RANSAC. And we can obtain the left boundary and right boundary of the checkerboard from the starting points and ending points on the scan lines of the plane point cloud.

In the last step, we split the left boundary and right boundary into four boundaries from detecting the turning point of the boundary, and we use line fitting again to filter out outliers to get the final 3D line parameters. The example result of planes and lines extraction is shown in Fig. 4.

D. Extrinsic Calibration of a Stereo Camera and a 3D LiDAR

In this section, we will describe the geometry constraints and extrinsic calibration algorithm of a stereo camera and a 3D LiDAR.

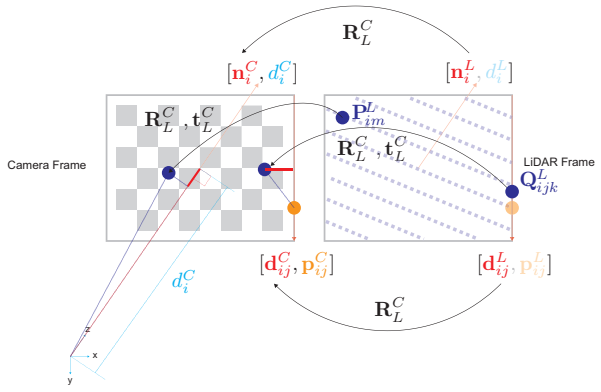


Fig. 5. Geometric Constraints

1) Geometric Constraints: The geometric constraints are illustrated in Fig. 5.

For line pairs $(L_{ij}^{C_1}, L_{ij}^{L})$ and $(L_{ij}^{C_2}, L_{ij}^{L})$, there are following constraints:

$$\mathbf{R}_L^{C_1}\mathbf{d}_{ij}^{L} = \mathbf{d}_{ij}^{C_1} \tag{9}$$

$$\mathbf{R}_L^{C_2}\mathbf{d}_{ij}^{L} = \mathbf{d}_{ij}^{C_2} \tag{10}$$

$$(\mathbf{I} - \mathbf{d}_{ij}^{C_1}(\mathbf{d}_{ij}^{C_1})^T)(\mathbf{R}_L^{C_1}\mathbf{Q}_{ijk}^{L} - \mathbf{p}_{ij}^{C_1} + \mathbf{t}_L^{C_1}) = \mathbf{0} \tag{11}$$

$$(\mathbf{I} - \mathbf{d}_{ij}^{C_2}(\mathbf{d}_{ij}^{C_2})^T)(\mathbf{R}_L^{C_2}\mathbf{Q}_{ijk}^{L} - \mathbf{p}_{ij}^{C_2} + \mathbf{t}_L^{C_2}) = \mathbf{0} \tag{12}$$

For plane pairs $(\pi_i^{C_1}, \pi_i^{L})$ and $(\pi_i^{C_1}, \pi_i^{L})$, there are following constraints:

$$\mathbf{R}_L^{C_1}\mathbf{n}_i^{L} = \mathbf{n}_i^{C_1} \tag{13}$$

$$\mathbf{R}_L^{C_2}\mathbf{n}_i^{L} = \mathbf{n}_i^{C_2} \tag{14}$$

$$\mathbf{n}_i^{C_1} \cdot (\mathbf{R}_L^{C_1}\mathbf{P}_{im}^{L} + \mathbf{t}_L^{C_1}) - d_i^{C_1} = 0 \tag{15}$$

$$\mathbf{n}_i^{C_2} \cdot (\mathbf{R}_L^{C_2}\mathbf{P}_{im}^{L} + \mathbf{t}_L^{C_2}) - d_i^{C_2} = 0 \tag{16}$$

In Eq. (9) (10), we use a rotation matrix between the LiDAR frame and the camera frame to transform the direction vector of the line from the LiDAR frame to the camera frame.

In Eq. (11) (12), we use rotation matrix and translation vector between the LiDAR frame and the camera frame to project $\mathbf{Q}_{ijk}^{L}$ which is one point on L_{ij}^{L} to the camera frame as $\mathbf{Q'}_{ijk}^{C}$, then connect it with $\mathbf{p}_{ij}$ to form the vector. And we use projection matrix $(\mathbf{I} - \mathbf{d}_{ij}^{C}(\mathbf{d}_{ij}^{C})^T)$ to project it to the vector from $\mathbf{Q'}_{ijk}^{C}$ to the line L_{ij}^{C}. This vector should have zero length since the projected point $\mathbf{Q'}_{ijk}^{C}$ should be also on the line L_{ij}^{C} in the camera frame.

In Eq. (13) (14), we use a rotation matrix between the LiDAR frame and the camera frame to transform the normal vector of the plane from the LiDAR frame to the camera frame.

In Eq. (15) (16), we use rotation matrix and translation vector between the LiDAR frame and the camera frame to project $\mathbf{P}_{im}^{L}$ which is one point on π_i^{L} to the camera frame as $\mathbf{P'}_{im}^{C}$, then project it on the normal vector $\mathbf{n}_i^{C}$ and minus d_i^{C} to get the distance to the plane π_i^{C}. Since the projected point $\mathbf{P'}_{im}^{C}$ should be also on the plane π_i^{C} in the camera frame, that distance to the plane π_i^{C} should have zero length.

2) Extrinsic Calibration from one pose: Using the above geometric constraints, [13] proves that we can compute the relationship between a 3D LiDAR and a monocular camera using 2 pairs of non-parallel line correspondences and 1 pair of plane correspondence with one pose. For the extrinsic calibration of a stereo visual system and 3D LiDAR, we can calibrate the 3D LiDAR with each camera to get the extrinsic parameters of the whole system.

3) Extrinsic Calibration from multiple poses: We can get a more accurate result using multiple poses. [13] introduced the extrinsic calibration method between a camera and a 3D liDAR from multiple poses using non-linear optimization. We can also formulate our problem as a non-linear optimization problem by minimizing line reprojection error and plane reprojection error from the 3D liDAR to the 3D camera to refine the extrinsic calibration result. And we can utilize LiDAR disparity constraint obtained by stereo calibration of the stereo visual system to optimize the extrinsic parameters globally.

Firstly, we minimize the following cost function to solve the initial $\mathbf{R}_L^{C_1}$ and $\mathbf{R}_L^{C_2}$:

$$\mathbf{R}_L^{C_1} = \arg\min_{\mathbf{R}_L^{C_1}} \sum_{i=1}^{N}\sum_{j=1}^{4} \|\mathbf{R}_L^{C_1}\mathbf{d}_{ij}^{L} - \mathbf{d}_{ij}^{C_1}\|^2 + \|\mathbf{R}_L^{C_1}\mathbf{n}_i^{L} - \mathbf{n}_i^{C_1}\|^2 \tag{17}$$

$$\mathbf{R}_L^{C_2} = \arg\min_{\mathbf{R}_L^{C_2}} \sum_{i=1}^{N}\sum_{j=1}^{4} \|\mathbf{R}_L^{C_2}\mathbf{d}_{ij}^{L} - \mathbf{d}_{ij}^{C_2}\|^2 + \|\mathbf{R}_L^{C_2}\mathbf{n}_i^{L} - \mathbf{n}_i^{C_2}\|^2 \tag{18}$$

As described in [28], the above problem can be solved by Singular Value Decomposition(SVD). We can define:

$$\mathbf{M}^{L} = [\mathbf{n}_1^{L}, \mathbf{d}_{11}^{L}, \ldots, \mathbf{d}_{14}^{L}, \ldots, \mathbf{n}_N^{L}, \mathbf{d}_{N1}^{L}, \ldots, \mathbf{d}_{N4}^{L}] \tag{19}$$

$$\mathbf{M}^{C_1} = [\mathbf{n}_1^{C_1}, \mathbf{d}_{11}^{C_1}, \ldots, \mathbf{d}_{14}^{C_1}, \ldots, \mathbf{n}_N^{C_1}, \mathbf{d}_{N1}^{C_1}, \ldots, \mathbf{d}_{N4}^{C_1}] \tag{20}$$

$$\mathbf{M}^{C_2} = [\mathbf{n}_1^{C_2}, \mathbf{d}_{11}^{C_2}, \ldots, \mathbf{d}_{14}^{C_2}, \ldots, \mathbf{n}_N^{C_2}, \mathbf{d}_{N1}^{C_2}, \ldots, \mathbf{d}_{N4}^{C_2}] \tag{21}$$

Using SVD, we can get

$$M^{L}(M^{C_1})^T = U_1\Sigma_1V_1^T \tag{22}$$

$$M^{L}(M^{C_2})^T = U_2\Sigma_2V_2^T. \tag{23}$$

Then, the rotation matrix $R_L^{C_1}$ and $R_L^{C_2}$ can be solved by closed-form solution:

$$R_L^{C_1} = V_1U_1^T \tag{24}$$

$$R_L^{C_2} = V_2U_2^T \tag{25}$$

To get intial translation vector $t_L^{C_1}$ and $t_L^{C_2}$, we need to utilize Eq. (11) (12) (15) (16). We denote $\bar{\mathbf{P}}_i^{L}$ as the centroids of the planes $\mathbf{P}_i^{L}$ in the LiDAR frame, $\bar{\mathbf{Q}}_{ij}^{L}$ as the entroids of the planes $\mathbf{Q}_{ij}^{L}$ in the LiDAR frame. Using above definitions and constraints, we can have

$$\mathbf{n}_i^{C_1} \cdot \mathbf{t}_L^{C_1} = -\mathbf{n}_i^{C_1} \cdot \mathbf{R}_L^{C_1} \bar{\mathbf{P}}_i^L + d_i^{C_1} \tag{26}$$

$$\mathbf{n}_i^{C_2} \cdot \mathbf{t}_L^{C_2} = -\mathbf{n}_i^{C_2} \cdot \mathbf{R}_L^{C_2} \bar{\mathbf{P}}_i^L + d_i^{C_2} \tag{27}$$

$$\mathbf{A}_{ij}^{C_1} \mathbf{t}_L^{C_1} = -\mathbf{A}_{ij}^{C_1} (\mathbf{R}_L^{C_1} \bar{\mathbf{Q}}_{ij}^L - \mathbf{p}_{ij}^{C_1}) \tag{28}$$

$$\mathbf{A}_{ij}^{C_2} \mathbf{t}_L^{C_2} = -\mathbf{A}_{ij}^{C_2} (\mathbf{R}_L^{C_2} \bar{\mathbf{Q}}_{ij}^L - \mathbf{p}_{ij}^{C_2}) \tag{29}$$

We can solve this least-square problem to get a closed-form solution of $\mathbf{t}_L^{C_1}$ and $\mathbf{t}_L^{C_2}$.

After having the initial rotation matrices $\mathbf{R}_L^{C_1}$, $\mathbf{R}_L^{C_2}$ and the translation vectors $\mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}$, we can solve a non-linear optimzation problem by Levenberg-Marquardt algorithm[26].

Considering Eq. (11) (12) (15) (16) as constraints, we can formulate following cost function terms which describe the constraits between 3D LiDAR and the cameras:

$$\begin{aligned}\mathbf{e}_L^{C_1} = &\sum_{i=1}^{N} \frac{1}{N_i} \|\mathbf{n}_i^{C_1} \cdot (\mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}) - d_i^{C_1}\|^2 \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \|\mathbf{A}_{ij}^{C_1} (\mathbf{R}_L^{C_1} \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^{C_1} + \mathbf{t}_L^{C_1})\|^2\end{aligned} \tag{30}$$

$$\begin{aligned}\mathbf{e}_L^{C_2} = &\sum_{i=1}^{N} \frac{1}{N_i} \|\mathbf{n}_i^{C_2} \cdot (\mathbf{R}_L^{C_2} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_2}) - d_i^{C_2}\|^2 \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \|\mathbf{A}_{ij}^{C_2} (\mathbf{R}_L^{C_2} \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^{C_2} + \mathbf{t}_L^{C_2})\|^2\end{aligned} \tag{31}$$

We jointly optimize two pairs of extrinsic calibration parameters between each camera and 3D LiDAR by minimizing the following cost function:

$$(\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}) = \underset{\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}}{\arg\min} \quad \mathbf{e}_L^{C_1} + \mathbf{e}_L^{C_2} \tag{32}$$

Extrinsic calibration of a stereo visual system has been explored by many researchers, we can obtain acure extrinsic parameters including rotation matrix and translation vector between two cameras using multiple images containing checkerboard. We adopt method [29] to calibrate two cameras to get accurate extrinsic parameters $(\mathbf{R}_{C_1}^{C_2}, \mathbf{t}_{C_1}^{C_2})$ of the stereo visual system.

Since the extrinsic calibration between a camera and a 3D LiDAR has estimation error, we can use LiDAR disparity constraint to consider this estimation error with stereo calibration result. This LiDAR disparity constraint is illustrated in Fig. 6. We project the point cloud of the checkerboard plane $\{\mathbf{P}_{im}^L\}$ in the LiDAR frame to the left camera frame as $\{\mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}\}$ using estimated $(\mathbf{R}_L^{C_1}, \mathbf{t}_L^{C_1})$. Then we project $\{\mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}\}$ to right camera frame as $\{\mathbf{R}_{C_1}^{C_2}(\mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}) + \mathbf{t}_{C_1}^{C_2}\}$using stereo extrinsic parameters $(\mathbf{R}_{C_1}^{C_2}, \mathbf{t}_{C_1}^{C_2})$ obtained by stereo camera calibration. This projected point cloud has disparity compared to $\{\mathbf{R}_L^{C_2} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_2}\}$ projected from LiDAR frame to right camera frame using $(\mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_2})$. This LiDAR disparity can be described as

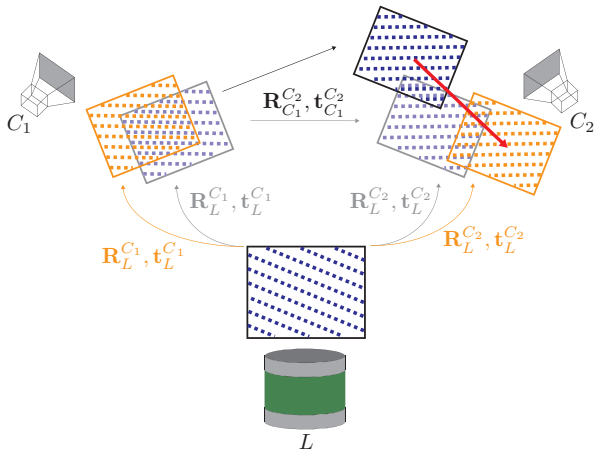


Fig. 6. LiDAR disparity constraint

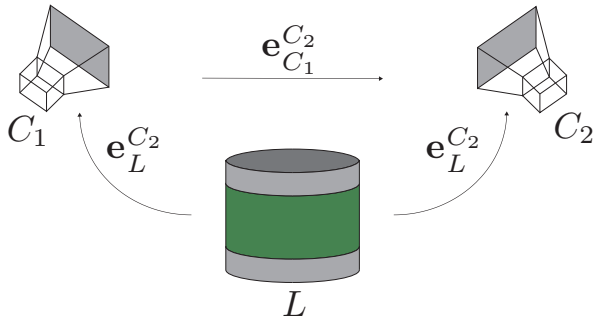


Fig. 7. Joint Optimization

$$\begin{aligned}\mathbf{e}_{C_1}^{C_2} = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{m=1}^{N_i} &\|\mathbf{R}_{C_1}^{C_2}(\mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}) + \mathbf{t}_{C_1}^{C_2} \\ &- (\mathbf{R}_L^{C_2} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_2})\|^2\end{aligned} \tag{33}$$

Combining LiDAR disparity constraint with Eq. (11) (12) (15) (16) we can joinly optimize the extrinsic parameters by minimizing the following cost function which is also illustrated in Fig. 7:

$$(\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}) = \underset{\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}}{\arg\min} \quad \mathbf{e}_L^{C_1} + \mathbf{e}_L^{C_2} + \mathbf{e}_{C_2}^{C_1} \tag{34}$$

Since the measurement error of the grid size of the checkerboard will influence the extrinsic calibration result of a camera and 3D LiDAR, [13] introduced similarity transformation to replace the rigid transformation. We also apply similarity transformation to extrinsic calibration between a stereo visual system and 3D LiDAR.

We introduce a scale factor s to refine the scale factor for the laser beam of the 3D LiDAR which is used for getting actual distances of points. After getting initial $\mathbf{R}_\mathbf{L}^{\mathbf{C_1}}$ and $\mathbf{R}_\mathbf{L}^{\mathbf{C_2}}$

in the same way, we can solve the linear system which is showed as below to get initial s, $\mathbf{t}_L^{C_1}$ and $\mathbf{t}_L^{C_2}$.

$$\mathbf{n}_i^{C_1} \cdot \mathbf{t}_L^{C_1} + \mathbf{n}_i^{C_1} \cdot \mathbf{R}_L^{C_1} \bar{\mathbf{P}}_i^L s = -d_i^{C_1} \tag{35}$$

$$\mathbf{n}_i^{C_2} \cdot \mathbf{t}_L^{C_2} + \mathbf{n}_i^{C_2} \cdot \mathbf{R}_L^{C_2} \bar{\mathbf{P}}_i^L s = -d_i^{C_2} \tag{36}$$

$$\mathbf{A}_{ij}^{C_1} \mathbf{t}_L^{C_1} + \mathbf{A}_{ij}^{C_1} \mathbf{R}_L^{C_1} \bar{\mathbf{Q}}_{ij}^L s = \mathbf{A}_{ij}^{C_1} \mathbf{p}_{ij}^{C_1} \tag{37}$$

$$\mathbf{A}_{ij}^{C_2} \mathbf{t}_L^{C_2} + \mathbf{A}_{ij}^{C_2} \mathbf{R}_L^{C_2} \bar{\mathbf{Q}}_{ij}^L s = \mathbf{A}_{ij}^{C_2} \mathbf{p}_{ij}^{C_2} \tag{38}$$

To refine the result, the cost function without considering the LiDAR disparity constraint is showed as below:

$$\begin{aligned}
&(s, \mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}) = \underset{\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}}{\arg\min} \\
&\sum_{i=1}^{N} \frac{1}{N_i} \| \mathbf{n}_i^{C_1} \cdot (s \mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}) - d_i^{C_1} \|^2 \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \| \mathbf{A}_{ij}^{C_1} (s \mathbf{R}_L^{C_1} \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^{C_1} + \mathbf{t}_L^{C_1}) \|^2 \\
&+ \sum_{i=1}^{N} \frac{1}{N_i} \| \mathbf{n}_i^{C_2} \cdot (s \mathbf{R}_L^{C_2} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_2}) - d_i^{C_2} \|^2 \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \| \mathbf{A}_{ij}^{C_2} (s \mathbf{R}_L^{C_2} \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^{C_2} + \mathbf{t}_L^{C_2}) \|^2
\end{aligned} \tag{39}$$

the cost function considering LiDAR disparity constraint is showed as below:

$$\begin{aligned}
&(s, \mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}) = \underset{\mathbf{R}_L^{C_1}, \mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_1}, \mathbf{t}_L^{C_2}}{\arg\min} \\
&\sum_{i=1}^{N} \frac{1}{N_i} \| \mathbf{n}_i^{C_1} \cdot (s \mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}) - d_i^{C_1} \|^2 \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \| \mathbf{A}_{ij}^{C_1} (s \mathbf{R}_L^{C_1} \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^{C_1} + \mathbf{t}_L^{C_1}) \|^2 \\
&+ \sum_{i=1}^{N} \frac{1}{N_i} \| \mathbf{n}_i^{C_2} \cdot (s \mathbf{R}_L^{C_2} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_2}) - d_i^{C_2} \|^2 \\
&+ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \| \mathbf{A}_{ij}^{C_2} (s \mathbf{R}_L^{C_2} \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^{C_2} + \mathbf{t}_L^{C_2}) \|^2 \\
&+ \sum_{i=1}^{N} \frac{1}{N_i} \sum_{m=1}^{N_i} \| \mathbf{R}_{C_1}^{C_2} (s \mathbf{R}_L^{C_1} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_1}) + \mathbf{t}_{C_1}^{C_2} \\
&- (s \mathbf{R}_L^{C_2} \mathbf{P}_{im}^L + \mathbf{t}_L^{C_2}) \|^2
\end{aligned} \tag{40}$$

Then we also jointly optimize the extrinsic parameters by minimizing the cost function using the Levenberg-Marquardt algorithm to refine the result.

IV. Experiments and Results

To evaluate our method, we compare our method to Unnikrishnan's method [9] which only used the plane information to calibrate a camera and a 3D LiDAR. Our experiment uses a Velodyne VLP-16 LiDAR (16 scan lines, $\pm 3cm$ range error, 360° horizontal field of view and ±15° vertical field of view)and a ZED stereo camera(1280 × 720 resolution) to evaluate our method.

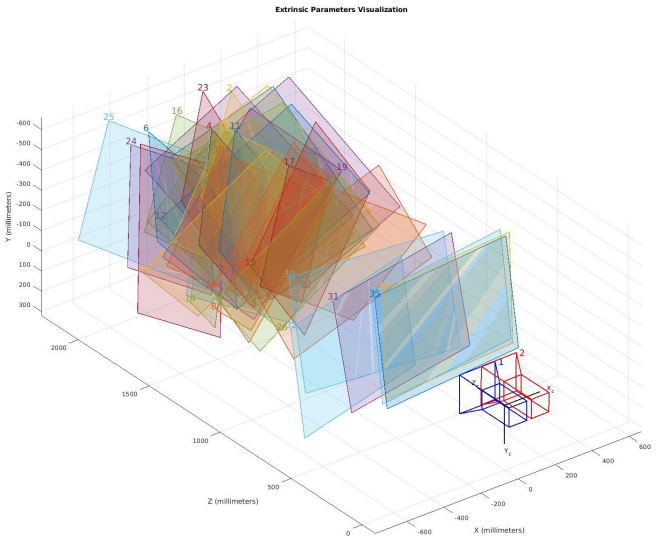


Fig. 8. Poses of checkerboards for the Stereo Calibration

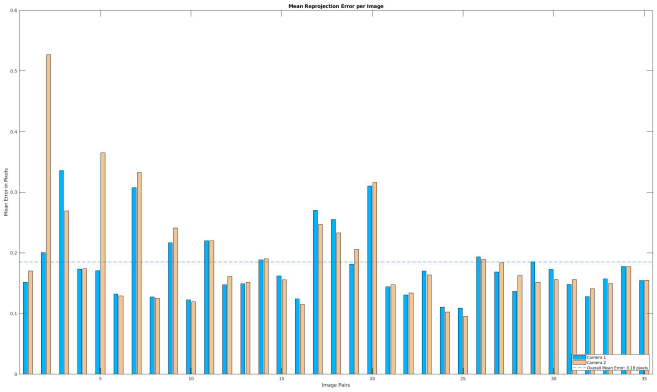


Fig. 9. Reprojection Error of the Stereo Calibration

We use stereo calibration with 35 poses to get accurate extrinsic parameters $(\mathbf{R}_{C_1}^{C_2}, \mathbf{t}_{C_1}^{C_2})$ of the stereo visual system. The poses of the checkerboards are showed in Fig. 8. The reprojection error of the stereo calibration is shown in Fig. 9.

To evaluate methods, we estimate the extrinsic parameters between each camera and 3D LiDAR $(\mathbf{R}_L^{C_1}, \mathbf{t}_L^{C_1}), (\mathbf{R}_L^{C_2}, \mathbf{t}_L^{C_2})$, then calculate the relative pose from left camera frame to right camera frame using these two transformation. For similarity transformation, we will regard it as the rigid transformation when we calculate the error.

To evaluate the transformation error between the estimated transformation $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ and the groundtruth transformation $(\mathbf{R}, \mathbf{t})$, we use $\frac{\|\hat{t}-t\|_2}{\|t\|_2}$ as the translation error and the angle-axis representation of $\hat{\mathbf{R}}\mathbf{R}^{-1}$ [30] as the rotation error.

We collected 30 pairs of LiDAR point clouds and images

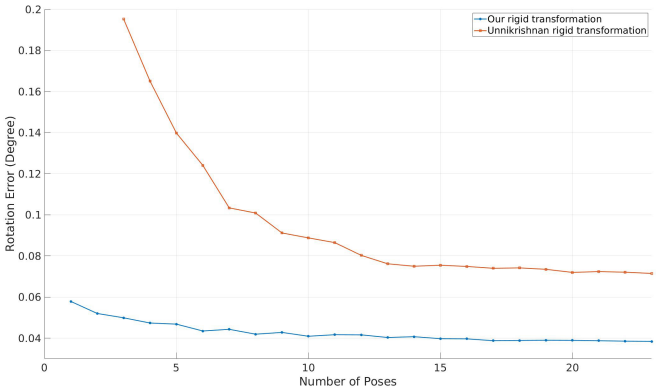


Fig. 10. Mean Rotation Error of rigid transformation, compare between our method without LiDAR disparity constraint and Unnikrishnn's algorithm[9]

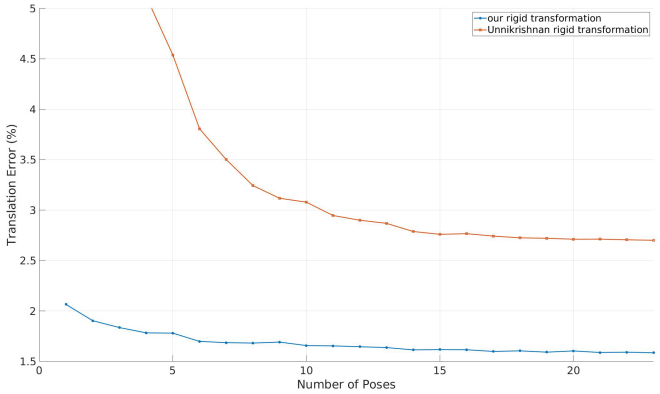


Fig. 11. Mean Translation Error of rigid transformation, compare between our method without LiDAR disparity constraint and Unnikrishnn's algorithm[9]

to do our evaluation, we choose $N \in [1, 23]$ randomly from them for our method, and $N \in [3, 23]$ for Unnikrishnan's method. We run the experiment 200 times for each $N \in [2, 25]$. We estimate extrinsic parameters for all 30 poses for $N = 1$.

The comparison of our method and Unnikrishnan's method is shown in Fig. 10 and Fig. 11. In this comparison experiment, we both use rigid transformation without using LiDAR disparity constraint to calibrate each camera with 3D LiDAR. The results show that our method significantly outperforms Unnikrishnan's method. And the results prove that our method can provide accurate extrinsic calibration result even with one pose.

We compare the result of rigid transformation and similarity transformation in different ways. Firstly, we use one pose to calibrate each camera with 3D LiDAR using rigid transformation and similarity transformation without using LiDAR disparity constraint. Secondly, we back-project the point cloud of the checkerboard plane from the LiDAR frame to the camera frame. The fusion result is showed in Fig. 12. We can observe from the fusion result that the similarity transformation provides a more accurate result that the rigid transformation.

We also show that the LiDAR disparity constraint can improve the extrinsic calibration result significantly by comparing the performance of rigid/similarity transformation with/without LiDAR disparity constraint. And the result shows that similarity transformation gives more improvement in translation estimation than rotation estimation. The rotation error and translation error is showed in Fig. 13 and Fig. 14.

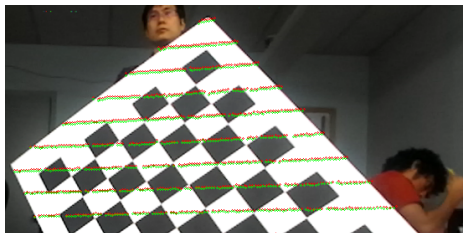


Fig. 12. Back-projection of the point cloud of the checkerboard plane using extrinsic parameters obtained from our rigid transformation method without LiDAR disparity constraint(red) and our similarity transformation method without LiDAR disparity constraint(green). Both methods use one pose.

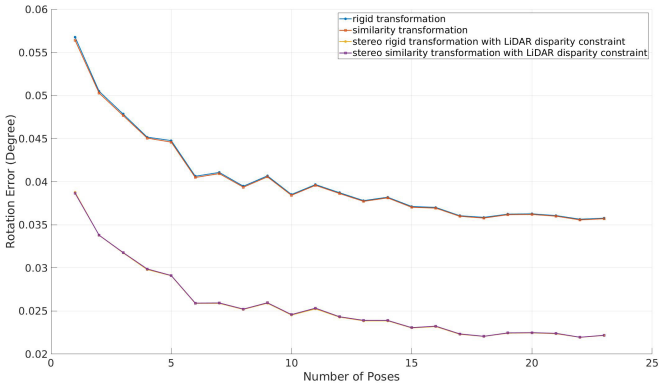


Fig. 13. Mean Rotation Error of our method: rigid/similarity transformation with/without LiDAR disparity constraint

V. Conclusions and Future Works

Using line and plane correspondences, we can use fewer (even one) poses to get more accurate extrinsic calibration result of a stereo visual system and 3D LiDAR. Using LiDAR disparity constraint, we can improve results with multiple poses significantly. We show that similarity transformation which does not require measurement of grid size of checkerboard can have more accurate result than rigid

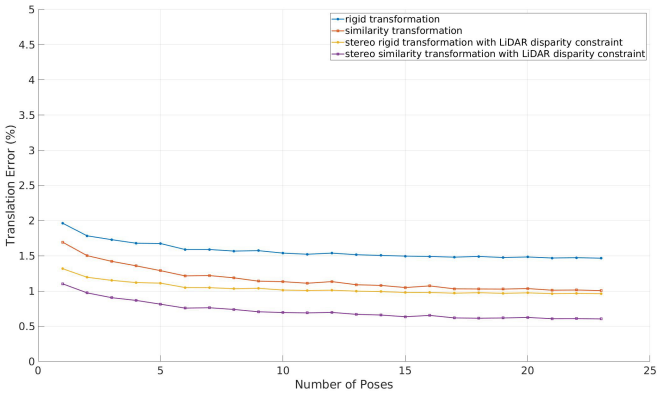


Fig. 14. Mean Translation Error of our method: rigid/similarity transformation with/without LiDAR disparity constraint

transformation. We implemented a calibration toolbox in C++ with high efficiency and accuracy. In the future, We will extend the calibration framework to multi-sensor calibration problem including intrinsic and extrinsic calibration.

ACKNOWLEDGMENT

Thanks to my mentor Lipu Zhou, Michael Kaess for guidance. Thanks to ShanghaiTech University for funding this work. Special Thanks to Dr. John Dolan, Ms. Rachel Burcin and the RISS team for support.

REFERENCES

[1] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: Science and Systems*, vol. 2, 2013.

[2] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information." in *AAAI*, 2012.

[3] Z. Taylor and J. Nieto, "Automatic calibration of lidar and camera images using normalized mutual information," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013.

[4] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Choosing a time and place for calibration of lidar-camera systems," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4349–4356.

[5] X. Gong, Y. Lin, and J. Liu, "3d lidar-camera extrinsic calibration using an arbitrary trihedron," *Sensors*, vol. 13, no. 2, pp. 1902–1918, 2013.

[6] V.-D. Hoang, D. C. Hernandez, H.-S. Park, and K.-H. Jo, "Closed-form solution to 3d points for estimating extrinsic parameters of camera and laser sensor," in *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*. IEEE, 2014, pp. 1932–1937.

[7] K. Kwak, D. F. Huber, H. Badino, and T. Kanade, "Extrinsic calibration of a single line scanning lidar and a camera," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3283–3289.

[8] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[9] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09*, 2005.

[10] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3936–3943.

[11] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic calibration of a 3d laser scanner and an omnidirectional camera," in *7th IFAC symposium on intelligent autonomous vehicles*, vol. 7, no. 1, 2010.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[13] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences," in *Intelligent Robots and Systems, 2018. IROS 2018. IEEE/RSJ International Conference on*. IEEE, 2018.

[14] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.

[15] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 4164–4169.

[16] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi, "Revisiting the pnp problem: A fast, general and optimal solution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2344–2351.

[17] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 1803–1810.

[18] H. Aliakbarpour, P. Nunez, J. A. Prado, K. Khoshhal, and J. Dias, "An efficient algorithm for extrinsic calibration between a 3d laser range finder and a stereo camera for surveillance." in *ICAR*, 2009, pp. 1–6.

[19] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017, pp. 1–6.

[20] Y. Li, Y. Ruichek, and C. Cappelle, "3d triangulation based extrinsic calibration between a stereo vision system and a lidar," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, 2011, pp. 797–802.

[21] V. John, Q. Long, Z. Liu, and S. Mita, "Automatic calibration and registration of lidar and stereo camera without calibration objects," in *Vehicular Electronics and Safety (ICVES), 2015 IEEE International Conference on*. IEEE, 2015, pp. 231–237.

[22] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Exploiting known unknowns: Scene induced cross-calibration of lidar-stereo systems," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 3647–3653.

[23] 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.

[24] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobbs journal of software tools*, 2000.

[25] K.-E. Jaeger, S. Ransac, B. W. Dijkstra, C. Colson, M. van Heuvel, and O. Misset, "Bacterial lipases," *FEMS microbiology reviews*, vol. 15, no. 1, pp. 29–63, 1994.

[26] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*. Springer, 1978, pp. 105–116.

[27] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[28] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 698–700, 1987.

[29] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[30] Z. Kukelova, J. Heller, and A. Fitzgibbon, "Efficient intersection of three quadrics and applications in computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1799–1808.

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS

Photo by: Terrence Ting