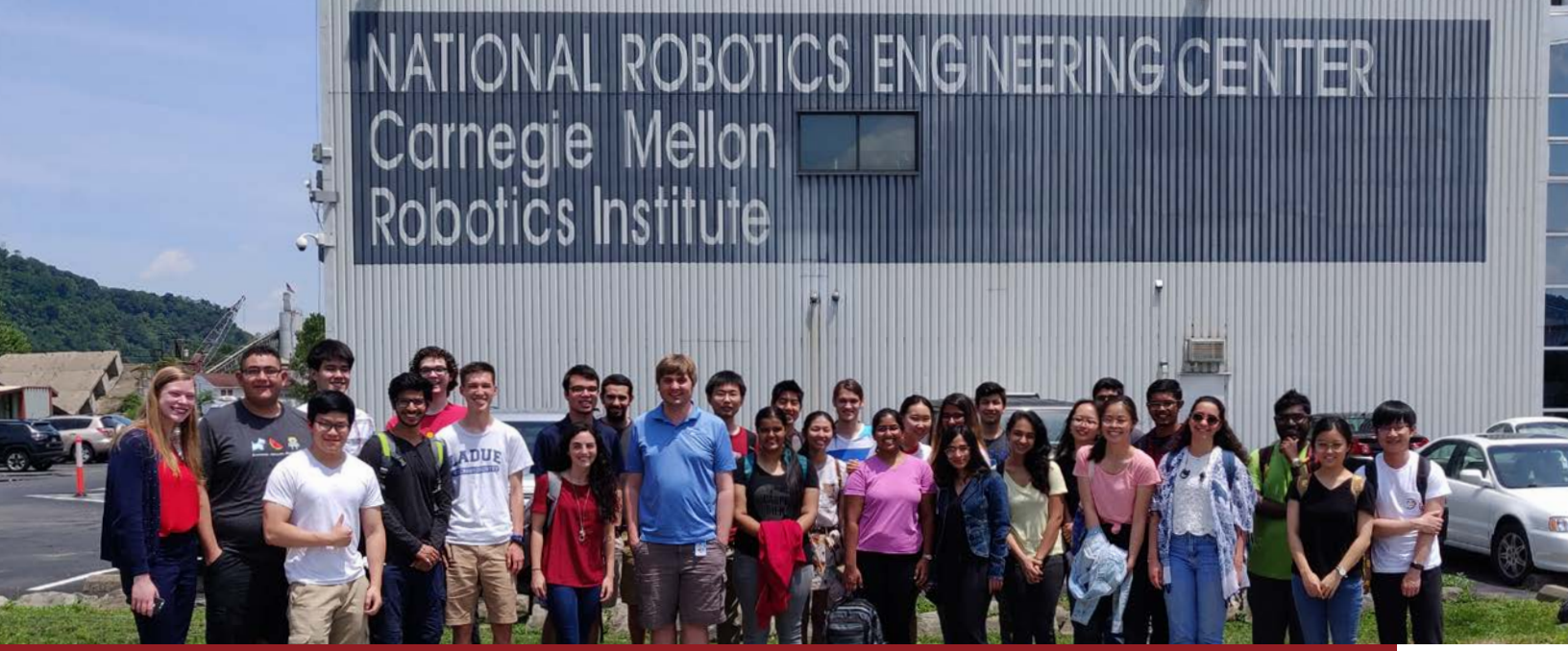
Carnegie Mellon University
The Robotics Institute

Working Papers **Journal**

Volume 7 | Fall 2019

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

2019 RISS Working Papers Scholar Editors
Quintessa Guengerich
Mononito Goswami

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

* *Chen's RISS participation in the 2019 cohort was under the title "RISS Program Intern," and not as a Robotics Institute Summer Scholar. This role involved a mixture of research and administrative support.*

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 Scholar Editors

The Robotics Institute Summer Scholars (RISS) program is an intensive summer program bringing together students from all over the world to participate in cutting-edge robotics research. The program culminates in the RISS Working Paper Journal, where each participating scholar shares the results and implications of their research.

Some scholars from past cohorts have gone on to be published in reputed conferences such as the International Conference on Robotics and Automation and International Conference on Intelligent Robots and Systems. For all scholars, publishing in the Working Paper Journal is a platform to showcase what was learned and accomplished, and to exchange ideas on future work.

This summer, the RISS cohort comprised of students from 7 countries, 14 US states and territories. Scholars answered a wide array of research questions this summer, ranging from robotics in healthcare and education, to intelligent transportation, path planning and scheduling.

Publishing in a Working Papers Journal requires not only a commitment to a research question, but also a commitment to clear communication and writing. The editors would like to thank the scholars and their mentors for the effort behind the works presented here.

Each paper in this journal underwent two rigorous rounds of peer review, and was reviewed by two scholars in a single-blind process. The journal team would like to thank Daniel Feshbach, Rebecca Martin, Patrick Naughton, Samuel Triest, Boshi Wang, and Nadia Atmulak for their participation in the peer review process. For their help with the journal's design details — such as the selection of photographs and templates — the editors would like to thank Rebecca Martin, Daniel Feshbach, Xinjie Yao and Ivana Collado. Other members of the journal team that the editors would like to thank are Sushant Wadavker, Valerie Chen, Sharmistha Gupta, and Andy Gao. Previous RISS alumni Stephanie Milani and David Russell also lent their time to guiding the peer review process; the team and editors would like to acknowledge and thank them as well.

The Global Communication Center at Carnegie Mellon also provide time this summer to help the cohort with their writing and presenting abilities. This came in the form of various workshops and many hours of individual appointments. Therefore, the editors would like to thank Emily Ferris, Alex Hall, Dr. Juliann Reineke, and Dr. Joanna Wolfe, of the GCC.

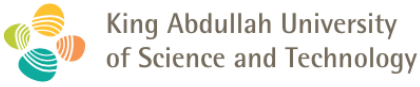
The RISS cohort would also like to thank the countless CMU faculty, staff, and graduate students who dedicated time to mentorship. What they give to the program is what allows summer students to learn from their research experience.

Finally, the RISS 2019 cohort would like to thank co-Directors Rachel Burcin and Dr. John Dolan. The effort that Rachel and John put into the RISS program is not only in the summer; it's year-round, as long as it takes to get the scholars where they want to be. This journal itself testifies to their dedication: the hard work and ambition of these scholars reflects a fraction of the effort that Rachel and John have put into this program.

Quintessa Guengerich and Mononito Goswami
2019 RISS Working Papers Scholar Editors

Thank You

At the core of the Carnegie Mellon University 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. We thank the RISS sponsors whose support is opening doors and creating futures in robotics & AI.



CMU Go Research Program

CMU Office of the Assistant Vice Provost for Graduate Education

CMU Global Communications Center

CMU Office of the Assistant Vice Provost for Educational Outreach

CMU Catering Services

Thank You Corporate Partners!

A Note of Thanks to DJI & UBTECH from the Entire 2019 RISS Cohort

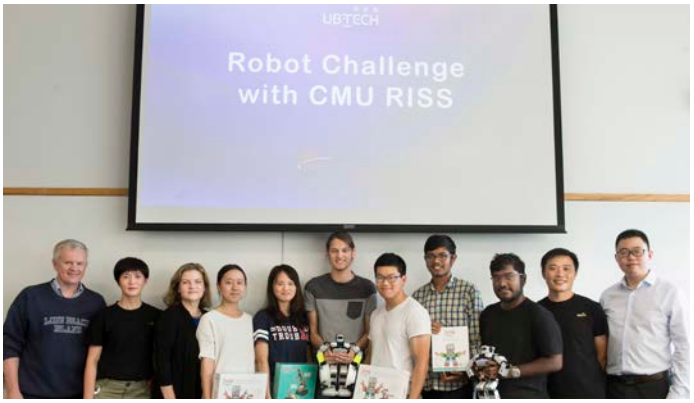
The Robotics Institute Summer Scholars offers research experience to students from around the world, working on cutting-edge research problems in computer vision, planning algorithms, and machine learning. However, a common request has reverberated through the Robotics Institute: how can so many skilled students never have worked with a real robot?

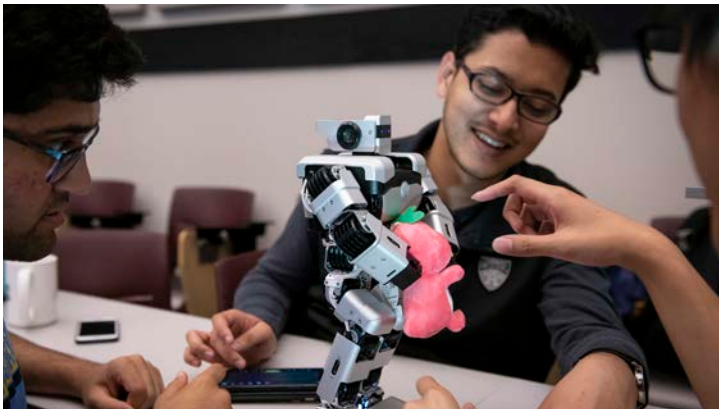
UBTECH and DJI Sciences and Technologies have provided tremendous support to the RISS program, and risen to address the shortage in hands-on robotics for summer scholars.

First, DJI joined the scholars and provided hand-held drones to complete an obstacle course. The challenge allowed scholars to use their skills in computer vision to see April Tags on the ground, then use knowledge in dynamics and controls to help the drone navigate to the next tag. For many students, the challenge was a first in computer vision, learning to have the drone look for the April Tag and recognize it. For other students, it was their first experience working with robot navigation and controls. The event allowed the students to gain exposure by working with a real drone, and help other scholars learn skills outside of their comfort zone.

Later in the summer, UBTECH came to Pittsburgh from China, and worked with scholars on a unique challenge. Using Yanshee humanoid robots and PS3 controllers, students were asked to "rescue animals" from danger zones and deliver them to safety. The challenge consisted of controls interfacing, kinematics, and style — and allowed students to apply their work on real robots. The results of the challenge were surprisingly creative: the fastest robot slithered on its back like a snake, and the best grabber was outfitted with gloves.

The Robotics Institute Summer Scholars of 2019 express their deep gratitude to UBTECH and DJI Sciences and Technologies for their incredible improvement upon the RISS experience. The time and hard work they gave to the scholars this summer is tangible in the skills we have taken forward with us.





THE FUTURE OF POSSIBLE

UBTECH EDUCATION

Semi-supervised Learning to Perceive Children's Affective States in a Tablet Tutor

Mansi Agarwal[1] and Jack Mostow[2]

Abstract— **Like good human tutors, intelligent tutoring systems should detect and respond to students' affective states. However, accuracy in detecting affective states automatically has been limited by the time and expense of manually labeling training data for supervised learning. To combat this limitation, we use semi-supervised learning to train an affective state detector on a sparsely labeled, culturally novel, authentic data set in the form of screen capture videos from a Swahili literacy and numeracy tablet tutor in Tanzania that shows the face of the child using it. We achieved 88% leave-1-child-out cross-validated accuracy in distinguishing pleasant, unpleasant, and neutral affective states, compared to only 61% for the best supervised learning method we tested. This work contributes toward using automated affect detection both offline to improve the design of intelligent tutors, and at runtime to respond to student affect based on input from a user-facing tablet camera or webcam.**

I. Introduction and Relation to Prior Work

The field of affective computing seeks to narrow the communicative gap between the naturally emotional human and the emotionally challenged computer by developing computational systems that recognize and respond to the affective states (*i.e.*, emotions) of the user [1]. In particular, considerable work has investigated the automated estimation of affective states from facial expressions (*e.g.* [2]) and other visual cues (*e.g.* [3]). Emotional expressions are socially reactive, so users may try to mask certain unpleasant emotions [4].

Much of the research on affective computing has focused on making intelligent tutoring systems react to students' emotions (*e.g.*, [5], [6], [7]). This paper likewise presents work on automated detection of children's affective states in an educational tablet app, RoboTutor. RoboTutor's thousands of activities teach basic literacy and numeracy to children who have little or no prior schooling. Our eventual goal for automated affect detection is to help RoboTutor increase children's engagement and learning gains. The work reported here is novel in several respects.

Novel population: Work on affect detection in intelligent tutors has typically focused on American high school and college students. In contrast, the work reported here is based on data from children ages 6-12 in Tanzania. This data is novel in three respects. First, few databases of facial expressions include children's faces [8] [9]. Second, even fewer include Africans [10]. Finally, emotional expression varies from culture to culture [11], so affect detectors trained on an American population might not work for an East African population.

Authentic context: Foundational research on emotion detection has mainly focused on six "basic" emotions (happiness, sadness, surprise, disgust, anger, and fear) [12]. Typically these emotions are represented by deliberate facial expressions [13] or elicited by experimental stimuli [14]. In contrast, the affective states relevant to intelligent tutors are students' normal reactions to them, namely boredom, confusion, delight, frustration, surprise, and neutral or "flow" [15]. Most facial expression data is recorded in well-controlled laboratory conditions. In contrast, this paper is based on data from authentic contexts of children using RoboTutor.

Camera-only: Even data on authentic affective states in intelligent tutors are typically collected in heavily instrumented laboratory conditions using an expensive array of devices such as pressure sensors [16] and EEG headsets [17], as well as video from cameras external to the tutor itself. These input signals are informative for research but not practical outside the lab.

In contrast, we use only video input recorded by Google Pixel C Android tablets running RoboTutor in the field, both indoors and outdoors. Limiting its temporal and spatial resolution served to avoid filling up tablet memory or swamping the WiFi bandwidth required to send it to our lab for analysis. This data is therefore characterized by limited resolution, variable indoor and outdoor illumination, and occlusion by children's friends and their own hands.

Multi-channel: Facial expressions are an important visual channel to convey emotions, but by no means the only one. We also use other features known to reflect affective states: head proximity [18], head orientation [19], blink rate [20], pupil size [21], and eye gaze [3].

Semi-supervised: Systems that rely on supervised machine learning require large amounts of labeled training data (*e.g.* [22], [23]). Labeling affective states by hand is costly and time-consuming. Therefore we employ a semi-supervised approach for training an affective state detector on a sparsely labeled dataset [24]. That is, we train a classifier on the manually labeled instances, "pseudo-label" a subset of the unlabeled data using the trained classifier, retrain it on the expanded set of labeled data, repeat, and iterate.

In summary, this paper presents and evaluates a multi-channel affective state recognizer trained on authentic, partially labeled data from a novel population and context. The rest of the paper is organized as follows: Section II describes our data set. Section III specifies our methodology

for training the affective state detector. Section IV reports our results. Finally, Section V summarizes contributions, limitations, and future work.

II. Data Set

The data for the present study come from 229 screen capture videos of approximately 30 children using RoboTutor on two tablets in Tanzania between 6/22/2016 and 7/17/2017. Each video typically shows one session lasting 20 – 30 minutes (until the next child's turn). The videos were recorded by a free app called AZ Screen Recorder [25], which displayed the front-facing camera input in a small window and included it in the screen video it recorded, as shown in Figure 1. To limit storage consumption, we configured AZ Screen Recorder to record at a temporal resolution of 48 frames per second and a spatial resolution of 1024 x 720 pixels, of which the camera window took 192 x 148 pixels.

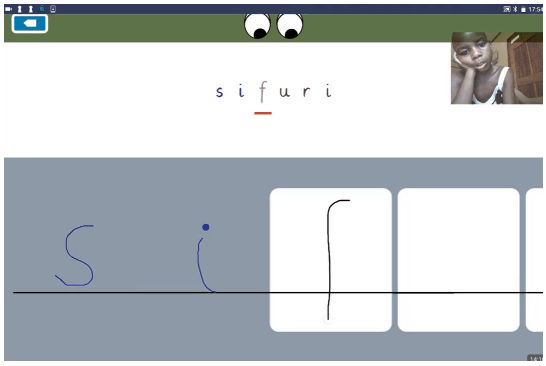


Fig. 1: RoboTutor interface with camera window.

The entire 100+ hours of video was far too large to label manually, so we selected approximately 345 short clips to label. As in previous work [26], we excluded the first and last minute of a video so as to avoid artifacts at the start and end of each session. Based on watching a few clips, we determined that 10-second clips were long enough to label yet short enough to be dominated by a single affective state.

We used two types of sampling to find clips to label. Based on previous research [27], we expected *neutral* to be by far the most frequent affective state. To find likely instances of less common affective states, we randomly selected seven of the videos, located unusually high or low values of various visual features, *i.e.*, local maxima or minima more than 3 standard deviations from the mean, and chose 10-second windows centered at these points. This method yielded 285 clips. To obtain an unbiased sample more representative of typical affective behavior, we randomly chose 60 10-second clips from 10 other videos. We randomly intermixed the two samples and partitioned them into 16 batches, each comprising 15-20 pairs of clips.

We constructed a separate Google form for each batch, with these instructions:

1) This Google form will present a series of pairs of RoboTutor screen video clips for you to annotate.
2) The first clip of each pair contains just the zoomed-in camera input showing the kid.
3) The other clip shows the entire screen, including the camera input. White dots indicate screen touches.
4) Pick the option that fits best. If it fits poorly, or if another option fits almost as well, use the Comments field to explain why.

This protocol first elicited judges' perception of the child's affective state based solely on the video clip of the child, without any additional context, and then based on the video showing the same time interval but in the context of the entire RoboTutor screen. Thus changes in label from the first clip to the second clip could reveal the influence of context on the judge's perception of the affective state.

There were two questions for each clip.

1) Is the student paying attention? *(Yes, No, Can't tell)*
2) Which of the following best describes the kid's state? *(Boredom, Confusion, Delight, Frustration, Neutral, Surprise, or I can't tell.)*

We expected the first question to be easy for both humans and computer to answer based simply on gaze, *i.e.* whether the child was looking at the screen. The second question required finer-grained distinctions, and in fact proved much harder. Figure 3 shows the six affective states manifested while the children were using RoboTutor.

To label our data, we recruited a Kenyan professor with PhDs in English and International Education, a Tanzanian with a PhD in Instructional Technology, a Tanzanian doctoral student in Linguistics, and two American undergraduate Psychology majors, all of whom were familiar with RoboTutor. It was important to include East African judges not only because they understood the Swahili spoken by RoboTutor and its users, but also because perception of affective states is known to be culture-dependent [11]. All the judges classified the clips independently.

A. Inter-rater reliability

As an intuitive measure of consistency in labeling, we computed the percentage agreement between the judges. To measure the degree to which it exceeded the amount of agreement expected by chance, we computed Cohen's Kappa κ.

To quantify the influence of cultural differences on annotation, we compared pairwise agreement between judges from similar backgrounds (East Africa or USA) versus agreement between judges from different backgrounds. The East African judges agreed 61% of the time, with κ of 0.58, compared to 55% and κ of 0.47 for the American judges, who averaged only 50% agreement with East African judges, with κ of 0.41. That is, judges agreed more often with judges from the same culture than with judges from another culture. Accordingly, we used only the East African judges' labels to train and test the classifier.

Judges agreed on some distinctions more than on others. In particular, they had trouble distinguishing frustration from confusion. One clue to the reason comes from the labeling protocol. The frequency with which judges changed their initial labels, which were based just on the camera input, reflects the extent to which they inferred affective states based at least in part on children's interactions with RoboTutor. Figure 2 shows the transition frequency from label i to label j, represented graphically by the arrow from i to j. The number of label changes was highest for frustration and confusion.

To avoid training our classifier on distinctions with low inter-rater reliability, we combined hard-to-distinguish states, thereby reducing the original set of 6 affective states to just 3 classes, namely *pleasant* (delight and surprise), *unpleasant* (boredom, confusion, and frustration), and *neutral* (flow). Inter-rater reliability was higher for this reduced set, with 67% agreement and κ of 0.63 on the cropped clips. Reliability was higher on the uncropped clips thanks to the additional context they provided, with 73% agreement and κ of 0.65. We used the cropped clips where both the judges agreed. These 231 "consensus" clips were distributed more equally among the 3 classes than among the original 6 affective states: 42 clips were labelled as *pleasant*, 91 as *unpleasant*, and 98 as *neutral*.

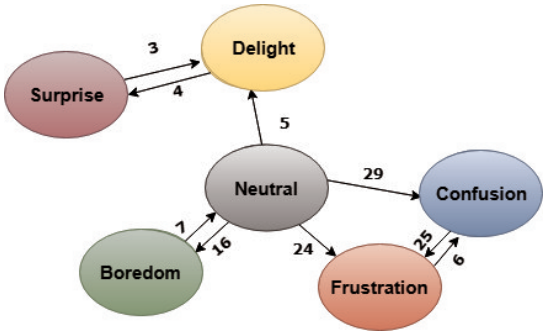


Fig. 2: Number of label changes from cropped to uncropped video clip

III. Approach

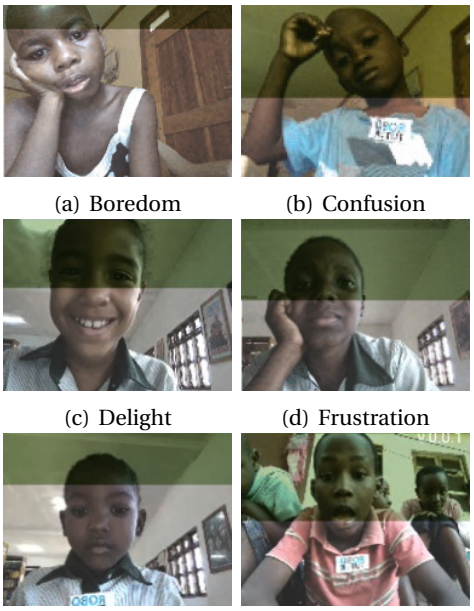
Our approach has 4 steps:

1) Extract features from the videos.
2) Aggregate each feature over the 10-second duration of a video clip into a single value.
3) Use semi-supervised learning to train a classifier on labeled and unlabeled data.
4) Use the trained classifier to predict the affective state of a child in a video clip.

We now describe each step in more detail.

A. Feature extraction

We started by extracting the camera input, which AZ Screen Recorder displayed over RoboTutor in a translucent window as shown in Figure 1. This window overlapped with the green banner at the top of the screen. Fortunately, this overlap did not prevent us from detecting faces and extracting useful information.



(a) Boredom (b) Confusion

(c) Delight (d) Frustration

(e) Neutral (f) Surprise

Fig. 3: The six affective states

As Figure 4 shows, this information consisted of visual features relevant to affective state, namely head proximity, head orientation, facial action units, blink rate, pupil size, and eye gaze. Each of these features provides a different channel of visual information. To help extract these features from video input, we used OpenFace [28], an open-source facial behavior analysis toolkit trained on a large collection of facial data sets, both static images and videos, diverse in age, gender, and ethnicity.

We now describe how we computed and used each feature.

Head proximity: Research on body language has shown that leaning forward indicates an increase in interest and leaning backward shows disinterest [18]. This finding motivated us to measure head distance to the camera. Openface gives the location of the head in millimeter coordinates as (H_x, H_y, H_z) in a 3-dimensional reference frame with the camera at the origin, where the X axis is horizontal, the Y axis is vertical, and the camera is pointed along the Z axis. We computed the Euclidean distance of the head from the camera as shown in Equation 1.

$$H_d = \sqrt{(H_x^2 + H_y^2 + H_z^2)} \quad (1)$$

Head orientation: OpenFace computes pitch (R_x), yaw (R_y), and roll (R_z) of the head rotation relative to the location and orientation of the camera. The rotation is in radians around the X, Y, and Z axes. When restless, humans tend to be more fidgety and hence move their heads unconsciously. Therefore, head orientation is the overall angle of the head from the baseline and reflects affective state. Equation 2 specifies head orientation as a

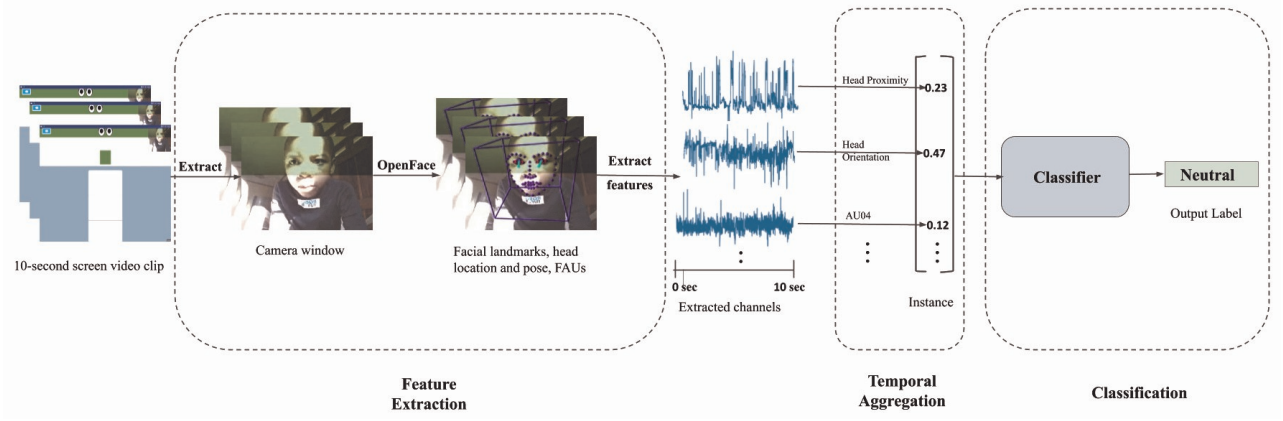


Fig. 4: System Architecture.

function of pitch, yaw, and roll.

$$H_o = R_x * R_y * R_z \tag{2}$$

Facial action units: Prior work on affective state recognizers has focused on Facial Action Units (FAUs) that were most diagnostic of the learning-centered emotions. Following [4], we employ AU04, AU07, AU12, AU25, AU26, and AU45. For every FAU, OpenFace outputs a classification and regression value. To obtain a continuous channel, we used the regression value, which measures the intensity of the FAU's presence as absent (value = 0), low (< 0.2), medium (0.2 − 0.7), or high (> 0.7).

Blink rate: Researchers have found that when nervous or troubled, humans' blink rate increases [20]. Using the eye coordinates obtained from OpenFace, we calculated an eye-aspect ratio [20] for each eye and used the average value of both eyes. If the eye aspect ratio was below a threshold θ for t frames, we considered it to be a blink. We tried different values for these two thresholds. $\theta = 0.4$ and $t = 4$ gave the best accuracy on a sample of 10 video clips. Equation 3 formally defines the eye aspect ratio (E_r) as:

$$E_r = \frac{\|(h-b)\| + \|(f-d)\|}{2 * \|(e-a)\|} \tag{3}$$

where *a, b, d, e, f,* and *h* are eye landmark coordinates obtained from OpenFace (Fig. 5).

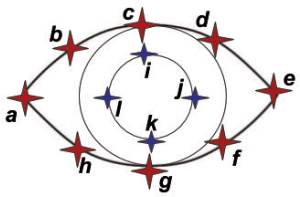


Fig. 5: Eye coordinates obtained from OpenFace

Pupil size: Pupil size reflects whether a person is aroused and alert, or bored and fatigued [29], so it is a useful indicator of affective state. Using the eye coordinates computed by OpenFace, we determined the ratio of the area of the pupil to the area of the eye using Equation 4. The ratio helped us deal with situations where the child was too close to the screen, leading to a large pupil size, without any role of affect. We used the average of this ratio for both eyes as an input to our classifier. We used Equation 4 to compute this ratio (P_r) as follows:

$$P_r = \frac{\|(l-j)\| * \|(k-i)\|}{\|(e-a)\| * \|(g-c)\|} \tag{4}$$

where *a, c, e, g, i, j, k,* and *l* are eye landmark coordinates obtained from OpenFace (Fig. 5).

Eye gaze: Eye gaze has been used by many researchers to detect alertness, attentiveness, and awareness [3]. OpenFace outputs the gaze direction averaged across the two eyes as *(gaze_angle_x, gaze_angle_y)* in radians. Looking from right to left changes *gaze_angle_x* from negative to positive; looking from up to down changes *gaze_angle_y* from negative to positive. Looking straight ahead is represented as zero for both *gaze_angle_x* and *gaze_angle_y*.

B. Temporal aggregation

To avoid the complications and computational cost of time series analysis [30], we reduced each feature of a clip to a single summary value. To aggregate discrete features, we simply counted the number of occurrences in the video clip and divided by its duration to obtain a rate, *e.g.* blinks per second. To aggregate continuous features, we experimented with several functions:

- To summarize the feature, we computed its mean over the 10-second window. However, this function can be distorted by outliers.
- To combat distortions caused by noise, especially outliers, we computed its median over the window. However, this function fails to capture significant events shorter than half the duration of the clip.
- To measure the spike caused by the main event in the clip, we computed the maximum value of the feature. However, this function can be distorted by outliers.

- To accentuate spikes while reducing distortion by outliers, we computed the root mean squared value. However, this function is invariant to scrambling the order.
- To select clips based on extreme values of features, i.e. local minima and maxima at least 3 or more standard deviations from the mean, we chose the 10-second clip centered around each extreme value. To focus on its central region, we weighted the mean and root mean squared by dividing the value at each point in the clip by its distance t from the midpoint of the clip (plus an offset o to prevent division by zero).

We then normalized each aggregate feature value v to the interval $[0,1]$ as $(v-min)/(max-min)$, where max and min are the largest and smallest aggregate values of the feature over the entire set of 10-second clips. We found that proximity-weighted root mean squared achieved the highest cross-validated accuracy when used in a classifier trained as we now describe.

C. Semi-supervised learning

Semi-supervised learning [24] trains a classifier by using unlabeled data to augment sparse labeled data in order to achieve higher classification accuracy. We use the following algorithm:

Algorithm 1 Semi-Supervised Learning

Initialize the training set T to the set of labeled instances.
while *unlabelled instances remain* **do**
  Train a classifier on T.
  Select an unlabelled instance.
  Run the classifier on it.
  if *classifier predicts class C with confidence* τ **then**
    Add the instance to the training set T with pseudo-label C.
  end
end

One can select an unlabelled instance at random, or choose the one closest to a labeled instance. For efficiency, we chose the 10 unlabeled instances closest to any of the labeled instances. In practice the method always exhausted all the unlabelled instances. In theory it could reach a state where it couldn't classify any of them with confidence τ, in which case it should terminate.

We considered several popular classifier learning methods. We chose Random Forest because it performed best on our data (see Table I). We computed the confidence of a prediction as the percentage of trees in the forest that predicted class C. We set our confidence threshold τ to 0.9.

IV. Results

To illustrate how our detector works in practice by using the features defined in Section III-A, Figure 6 shows an instance classified as a pleasant affective state. After struggling to write the preceding number *0*, the child wrote the number *1* correctly on the first try, and smiled when RoboTutor responded *Mzuri!* ("good" in Swahili). This clip had negligible deflection from typical head orientation. Eye gaze and blink rate were normal, *i.e.*, within one standard deviation of their respective means. However, the child was very close to the screen, *i.e.*, head distance was less than its mean value by more than three standard deviations. Head proximity typically indicates engagement. Also, his pupils were dilated, which typically indicates interest. AU04 (Brow Lowerer) and AU45 (Blink) were absent, AU25 (Lips Part) was present with low intensity, AU07 (Lid Tightener) and AU26 (Jaw Drop) were present with medium intensity, and AU12 (Lip Corner Puller) was present with high intensity. As this example illustrates, our detector's recognition of pleasant instances is probably influenced by head proximity, pupil size, and smiling. We say "probably" because a random forest's calculations are too complicated to readily analyze the individual influence of all the features. Instead, we described their values relative to their distributions, on the assumption that unusual values are likely to affect the classifier output.

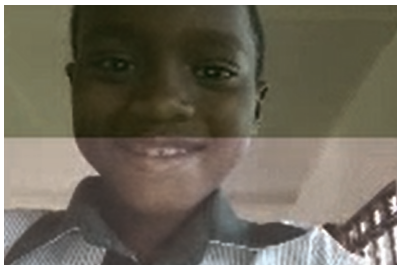


Fig. 6: Pleasant Instance

A. Quantitative Evaluation

To estimate performance on unseen children, we used leave-1-child-out cross-validation, training the classifier on all but one child and testing it on the held-out child. We performed this process for 5 randomly chosen children and report the median results. **Accuracy** is the percentage of test instances classified correctly. This measure is simplest and has practical significance because it predicts performance on unseen data drawn from the same distribution. However, it is sensitive to that distribution. In contrast, the following weighted measures are weight-averaged across all three classes, and therefore independent of the training set distribution. Each class is assigned a weight equal to the ratio of the number of instances in that class to the total number of instances in the test set. **Weighted precision** is the percentage correct among the instances classified as positive. **Weighted recall** is the percentage correct among the true positive instances. **Weighted F1** is the harmonic mean of weighted recall and weighted precision. As unlabelled data, we used 1007 clips from 20 videos, sampled using the same sampling techniques described in Section II.

As Table I shows, our method beat the supervised learning methods on all four criteria:

Classifier	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.21	0.52	0.21	0.23
Decision Tree	0.44	0.43	0.44	0.43
SVM	0.51	0.49	0.51	0.46
Adaboost	0.47	0.47	0.47	0.47
Logistic Regression	0.53	0.50	0.53	0.51
KNN	0.53	0.53	0.53	0.53
Random Forest	**0.61**	**0.63**	**0.61**	**0.60**
Proposed Approach	**0.88**	**0.88**	**0.84**	**0.86**

TABLE I: Comparison with supervised learning.

These results were for the consensus data where both judges agreed. We further tested our classifier on all our labeled test data, including 55 consensus clips not used for training and 114 clips on which the judges disagreed. This experiment helped us quantify the degradation in performance due to label noise. We evaluated the accuracy of the prediction compared to both judges' labels and took the average. As expected, average accuracy dropped from 88% to 56% when we included the non-consensus labels, compared to testing on the consensus data alone. The lower accuracy on the unfiltered data reflects the inherent difficulty of replicating subjective judgments on which human experts disagree.

To estimate the effect of cultural differences, we tested our trained classifier on both sets of consensus labels, African and American. Accuracy fell from 88% to 57% when tested on American labels. This difference quantifies the effect of cultural influence on people's facial expressions and other visual cues, and the consequent importance of recruiting judges from the same culture to label their affective states.

B. Error Analysis

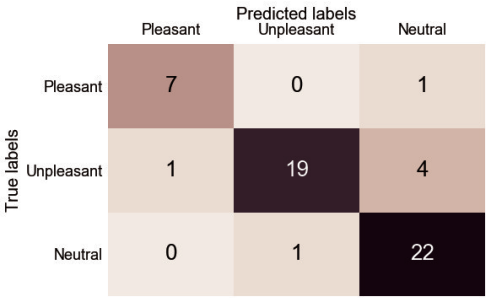


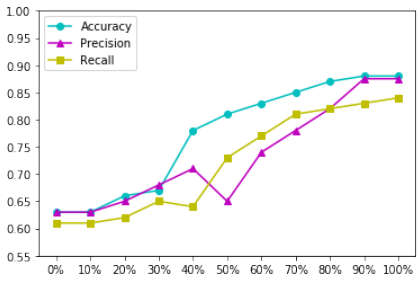
Fig. 7: Confusion Matrix

Figure 7 shows that our classifier performed well in most cases. However, the classifier incorrectly characterized four unpleasant instances as neutral. Most of these misclassifications involved boredom. Boredom is not easily distinguishable from neutral based on facial features. Indeed, boredom typically lacks facial expression. To detect boredom, we may have to use additional indicators, such as posture and acoustic-prosodic features of speech.

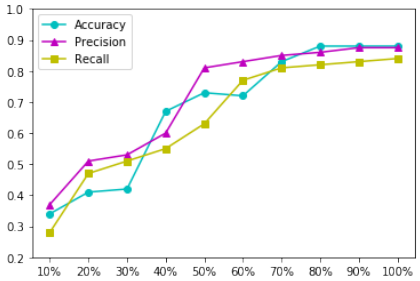
Accuracy is limited by the quality of the visual features input by the classifier from OpenFace, which depend in turn on its face detection. Our data come from authentic settings subject to varying illumination and occlusion. Consequently, OpenFace occasionally (especially in low-light conditions) fails to detect a face when it is present. Inspection of sample videos showed that OpenFace failed to detect a face approximately 2% of the time, typically for a second at a time.

C. Sensitivity analysis

To explore the sensitivity of the results to different factors, we varied the amount of unlabeled data, the amount of labeled data, and the method for selecting unlabeled data.



(a)



(b)

Fig. 8: Classifier Performance vs. (a) Number of Unlabeled Training Instances (b) Number of Labeled Training Instances

Effect of amount of unlabeled data: How did test accuracy vary with the amount of unlabeled data? We started with no unlabeled data, i.e., supervised learning, and added 10% of the unlabeled data at each iteration until all of the unlabeled data was utilized. Figure 8a shows that as the number of unlabeled instances increased from zero to 1007, accuracy rose asymptotically from 61% to 88%.

Effect of amount of labeled data: To analyze the effect of the amount of labeled data on classifier performance, we varied the percentage of labelled instances used from 10% to 100%, keeping the unlabeled training set constant, i.e. 1007 instances. Figure 8b shows that:

1) Accuracy, precision, and recall increased with the amount of labeled data, as expected.
2) Too little labeled data produced poor results, even with all the unlabeled data.

Effect of choice of data to pseudo-label: We conducted an experiment to understand why unlabeled data helped, and where the action was. We hypothesized that the order in which semi-supervised learning chose unlabeled instances to pseudo-label had a substantial effect on the performance of the resulting classifier. To test this hypothesis, we compared choosing 10 random instances at each iteration versus choosing the 10 instances closest to the instances labeled (or pseudo-labeled) so far.

Data pseudo-labeled	Accuracy	Precision	Recall	F1-Score
10 Random instances	0.78	0.78	0.72	0.74
10 closest instances	**0.88**	**0.88**	**0.84**	**0.86**

TABLE II: Effect of order of pseudo-labeling

Table II shows that choosing the 10 nearest instances at each iteration performed better than choosing 10 random instances. Why? Semi-supervised learning exploits the continuity assumption that instances near each other are likelier to belong to the same class than other instances assigned to that class based solely on generalization by a classifier trained on incomplete training data.

V. Conclusion

Contributions: We presented an innovative, multi-channel method for automating affect detection in a tablet app solely by integrating visual cues extracted from its front-facing camera input. We used semi-supervised learning to leverage our sparsely labeled training data. We evaluated it against human judges on authentic data from a novel population of children using the app in natural settings, and analyzed its performance both quantitatively and qualitatively. This work constitutes significant progress in automated affect detection, whether to improve tutor design off-line or to respond to student affect at runtime.

Limitations and future work: To increase inter-rater reliability, we combined confusable affective states into the same class. Future work to distinguish them could enhance RoboTutor's emotional intelligence.

The evaluated method is based solely on input from the tablet's front-facing camera, consistent with our focus on identifying what information about affect we can derive from visual cues. This type of input is more practical in realistic settings than inputs currently available in lab settings, such as EEG, pressure sensors, or even video from external cameras. However, some other types of input are readily available to a tablet tutor.

In particular, tablets input audio. Speech input is pedagogically informative when it can accurately be recognized or analyzed for other properties, such as prosody. Both these uses of audio input are feasible in quiet lab settings. However, in natural settings where multiple children use tablets in close proximity and noise-canceling headset microphones are too fragile or expensive, audio input is liberally contaminated with background speech from other children and their tablets.

The tutor itself could be a fruitful source of information, including its internal states, decisions, and actions, and student input such as screen taps and other gestures. Such data is tutor-specific but informative, and we plan to exploit it in the future, especially to recognize the contextual clues that our judges used to distinguish among boredom, confusion, and frustration.

We implemented our detector on a Windows PC. We can use it off-line to analyze screen-recorded sessions for guidance in redesigning RoboTutor. In principle, it could be applied to any screen capture video that includes camera input of the user, whether from the front-facing camera of a tablet, or the webcam atop a computer monitor.

However, RoboTutor itself runs on Android tablets. Incorporating the detector into RoboTutor will require porting it to an Android tablet to detect facial expressions and other visual cues in real time. We will also need to redesign RoboTutor to respond to detected affective states, evaluate the effects of such responses, and refine them accordingly. These responses should improve RoboTutor's ability to engage children and help them learn, and may generalize usefully to other tutors as well.

Acknowledgements

This work would not have been possible without the support of the RoboTutor team, especially Fortunatus Massewe for recording many hours of RoboTutor screen video, and Dr. Leonora Anyango Kivuva, Joash Gambarage, Sheba Naderzad, and Andrew McReynolds for labeling so much data. We would also like to thank Rachel Burcin, Dr. John Dolan and Mikayla Trost for tirelessly supporting us throughout the summer. The first author was a Robotics Institute Summer Scholar supported by an S.N. Bose Scholarship.

References

[1] R. W. Picard, *Affective computing*. MIT press, 2000.

[2] D. R. Faria, M. Vieira, F. C. Faria, and C. Premebida, "Affective facial expressions recognition for human-robot interaction," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017, pp. 805–810.

[3] J. Bidwell and H. Fuchs, "Classroom analytics: Measuring student engagement with automated gaze tracking," *Behav Res Methods*, vol. 49, p. 113, 2011.

[4] B. McDaniel, S. D'Mello, B. King, P. Chipman, K. Tapp, and A. Graesser, "Facial features for affective state detection in learning environments," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, no. 29, 2007.

[5] B. Woolf, W. Burleson, I. Arroyo, T. Dragon, D. Cooper, and R. Picard, "Affect-aware tutors: recognising and responding to student affect," *International Journal of Learning Technology*, vol. 4, no. 3-4, pp. 129–164, 2009.

[6] S. D'Mello and A. Graesser, "Autotutor and affective autotutor: Learning by talking with cognitively and emotionally intelligent computers that talk back," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 2, no. 4, p. 23, 2012.

[7] S. Craig, A. Graesser, J. Sullins, and B. Gholson, "Affect and learning: an exploratory look into the role of affect in learning with autotutor," *Journal of educational media*, vol. 29, no. 3, pp. 241–250, 2004.

[8] H. L. Egger, D. S. Pine, E. Nelson, E. Leibenluft, M. Ernst, K. E. Towbin, and A. Angold, "The nimh child emotional faces picture set (nimh-chefs): a new set of children's facial emotion stimuli," *International journal of methods in psychiatric research*, vol. 20, no. 3, pp. 145–156, 2011.

[9] B. Nojavanasghari, T. Baltrušaitis, C. E. Hughes, and L.-P. Morency, "Emoreact: a multimodal approach and dataset for recognizing emotional responses in children," in *Proceedings of the 18th acm international conference on multimodal interaction*. ACM, 2016, pp. 137–144.

[10] S. Du, Y. Tao, and A. M. Martinez, "Compound facial expressions of emotion," *Proceedings of the National Academy of Sciences*, vol. 111, no. 15, pp. E1454–E1462, 2014.

[11] D. Matsumoto, "Cultural influences on facial expressions of emotion," *Southern Journal of Communication*, vol. 56, no. 2, pp. 128–137, 1991.

[12] S. D. Craig, S. D'Mello, A. Witherspoon, and A. Graesser, "Emote aloud during learning with autotutor: Applying the facial action coding system to cognitive–affective states during learning," *Cognition and Emotion*, vol. 22, no. 5, pp. 777–788, 2008.

[13] K. Kaulard, D. W. Cunningham, H. H. Bülthoff, and C. Wallraven, "The mpi facial expression database—a validated database of emotional and conversational facial expressions," *PloS one*, vol. 7, no. 3, p. e32321, 2012.

[14] M. Valstar and M. Pantic, "Induced disgust, happiness and surprise: an addition to the mmi facial expression database," in *Proc. 3rd Intern. Workshop on EMOTION (satellite of LREC): Corpora for Research on Emotion and Affect*. Paris, France, 2010, p. 65.

[15] S. D'Mello, R. W. Picard, and A. Graesser, "Toward an affect-sensitive autotutor," *IEEE Intelligent Systems*, vol. 22, no. 4, pp. 53–61, 2007.

[16] S. D'Mello and A. Graesser, "Automatic detection of learner's affect from gross body language," *Applied Artificial Intelligence*, vol. 23, no. 2, pp. 123–150, 2009.

[17] P. C. Petrantonakis and L. J. Hadjileontiadis, "Emotion recognition from eeg using higher order crossings," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 186–197, 2009.

[18] D. Stanley, "Measuring attention using microsoft kinect," 2013.

[19] U. Hess, R. B. Adams, and R. E. Kleck, "Looking at you or looking elsewhere: The influence of head orientation on the signal value of emotional facial expressions," *Motivation and Emotion*, vol. 31, no. 2, pp. 137–144, 2007.

[20] Z. A. Haq and Z. Hasan, "Eye-blink rate detection for fatigue determination," in *2016 1st India International Conference on Information Processing (IICIP)*. IEEE, 2016, pp. 1–5.

[21] T. Partala and V. Surakka, "Pupil size variation as an indication of affective processing," *International journal of human-computer studies*, vol. 59, no. 1-2, pp. 185–198, 2003.

[22] P. Michel and R. El Kaliouby, "Real time facial expression recognition in video using support vector machines," in *Proceedings of the 5th international conference on Multimodal interfaces*. ACM, 2003, pp. 258–264.

[23] R. P. Reddy, P. M. Krishna, V. Narayanan, and S. Lalitha, "Affective state recognition using image cues," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 928–933.

[24] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[25] Hecorat. Az screen recorder (android). [Online; accessed 4-August-2019]. [Online]. Available: https://az-screen-recorder.en.uptodown.com/android

[26] J. K. Westlund, S. K. D'Mello, and A. M. Olney, "Motion tracker: Camera-based monitoring of bodily movements using motion silhouettes," *PloS one*, vol. 10, no. 6, p. e0130293, 2015.

[27] S. K. D'Mello and A. Graesser, "Multimodal semi-automated affect detection from conversational cues, gross body language, and facial features," *User Modeling and User-Adapted Interaction*, vol. 20, no. 2, pp. 147–187, 2010.

[28] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "Openface: an open source facial behavior analysis toolkit," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–10.

[29] M. E. Kret, "The role of pupil size in communication. is there room for learning?" *Cognition and Emotion*, vol. 32, no. 5, pp. 1139–1145, 2018.

[30] D. Ceballos and M. Sorrosal, "Time aggregation problems in financial time series," in *MS'2002 International Conference on Modelling and Simulation in Technical and Social Sciences*, 2002, pp. 25–27.

Incorporating Eye Gaze into Shared Autonomy for Assistive Robotics

Nadia AlMutlak[1] Reuben Aronson[2] and Henny Admoni[3]

***Abstract*—Robots enable people with disabilities to complete activities of daily living like eating independently. However, teleoperating robots in 3D can be challenging because of the robot's unintuitive kinematics and users' limited physical ability to provide input controls. Shared autonomy, a collaboration between autonomous robot control and high-level human control, was introduced to alleviate those challenges. Yet, shared autonomy only works with prior knowledge of the users' goals, and current goal prediction methods rely on explicit input from users to generate goal probabilities. Alternatively, eye gaze is a useful implicit input which can be used for predicting those goals given its ability to reveal people's intention and aid in collaborative activities. In this project, using activity specific gaze patterns such as alternating gaze fixations between a fork and plate while eating, we develop a model for gaze-based goal predictions. We then use previously collected data to create a two-step system that validates that model which: [1] interprets users' goals based on their fixations on potential targets, and [2] based on that interpretation, moves the robot's end-effector closer to that target. We intend to validate the system's effectiveness through a user study measuring task duration and user experience.**

***Index Terms*—Shared Autonomy, Assistive Robotics, Gaze Tracking, Goal Prediction**

I. Introduction

Robots are powerful enablers for people with whom they share spaces and collaborate to complete tasks that would otherwise be impossible. One instance of this is the collaboration between robots and people with disabilities in which robots mitigate the burden of completing activities of daily living such as eating or navigation. The integration of robotic systems into wheelchairs and robotic arms has allowed their users to complete tasks which were unmanageable due to their injuries or disabilities. For example, primarily used by people with spinal chord injuries, robotic arms like the Kinova Mico are controlled via joystick teleoperation to pour water or spear food with a fork. [1, 2]

However, teleoperating such robots is challenging and fatigues users. This may be attributed to the robots' unintuitive kinematics in a 3D environment since these robots have more degrees of freedom than a user can handle at a time. For example, robots like the Mico have 6 Degrees of Freedom (DOFs) which include (x, y, z, roll, pitch, yaw); whereas its controller, a 2D or 3D joystick, only has 2 DOFs (x, y) or 3 DOFs (x, y, z), respectively. As a result, users have to switch between modes to access different DOFs, which side tracks users and increases task time.[3] Challenges with teleoperation can be further exacerbated by the complicated motion planning needed by teleoperated systems which is limited by the user's range of motion. [4] Herlant et al have looked into solutions like automatic mode switching which enables autonomous shifts between the DOFs of the joystick for the user. They found there was no significant reduction in task time; however, through their survey they found that users preferred some robotic assistance navigating the task. [3] Further investigation revealed that users, in addition to not preferring fully manual teleoperation, did not prefer fully autonomous assistance either.[2, 3]

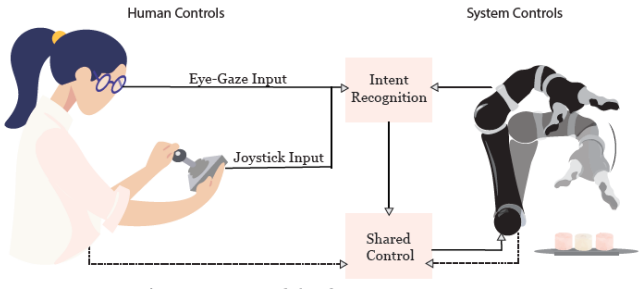


Fig. 1. Our proposed shared autonomy system incorporates gaze as an additional input for intent recognition. Both eye gaze input and joystick input are used to infer intent and generate a probability for the goal location. This probability is then input into the shared control algorithm which takes in both the user's real time control and the robot's control to continuously update the location of the goal and move the robot towards it.

Shared autonomy is presented as the happy medium of the two as it engages both the user's high-level task planning abilities and the robot's precision in a form of collaborative control to complete a task. This works in conjunction with prior knowledge of the user's goal to create a dynamic experience between the user and robot in which they continuously update the probability of a goal's location. Different adaptations of shared autonomy have collectively shown a relatively improved user experience.[2] For example, Javdani et al experimented with a food-spearing task on a robotic arm which evalauted multiple forms of control (full autonomy, shared autonomy, etc). They found that full autonomy was frustrating to users; whereas shared autonomy improved performance, requiring less physical effort and achieving greater precision with user's input [2].

While shared autonomy appears to be promising, it is still dependent on prior knowledge of the user's end goal which requires a large amount of information at the beginning of

[1]N. AlMutlak is Biomedical Engineering student at Columbia University in New York. nadia.almutlak@columbia.edu

[2]R. Aronson is a Ph.D. student in Carnegie Mellons Robotics Institute. rmaronson@cmu.edu

[3]H. Admoni is an Assistant Professor in Carnegie Mellons Robotics Institute and leads the Human and Robot Partners (HARP) lab. Henny@CMU.edu

a task [5]. Different control systems have approached their prediction framework in various ways that, while able to reduce task time, are not intuitive for users. A number of these systems can be described as trajectory-based prediction methods which rely on modal or coordinate changes from the input device to generate a probability distribution for a goal along a path. [2, 3] While these methods eventually locate the end goal, they are not able to provide assistance early in the task when assistance is most crucial.

Recent research has proposed that natural eye gaze may be used as an additional input for determining the user's end goal in conjunction with joystick teleoperation.[5, 6] Gaze has been shown to be a reliable form of non-verbal communication with the ability to convey a person's intentions. [7] Unlike joystick input, eye gaze is continuously provided by the user and does not require explicit input into a system. As a result of its implicit nature, gaze can be introduced into the shared autonomy system at no extra cost to the user to help the system infer the user's goal earlier in the task. Used as an additional input, gaze along with joystick input (as seen in Fig. 1) will infer more accurate goal predictions and create a seamless shared autonomy system that is more user friendly and time efficient.

In order to develop an intuitive shared autonomy system with the integration of eye-gaze-based predictions, this paper contributes the following:

1) The development and validation of a model which infers goals from eye gaze
2) A user study to measure the effectiveness of the integration of that model into the shared autonomy system using varied parameters

This work is currently in its developing stages, as such, this paper only presents the initial models developed for this project. However, we intend to run a user study and revisit the project in the future.

This paper is organized as follows: Section II discusses the work related to this project and provides crucial background information. Section III discusses our approach including the development of the models, their validation and selection. Section IV presents the suggested set up for our user study and future works for this project.

II. Related Work

This work builds on previously proposed approaches in shared autonomy, adapting the state-of-the-art shared autonomy systems presented by Jadvani et al. [2] It further explores work put forward by Admoni et al [5], which proposes gaze as a useful implicit input for goal predictions, by delving into manipulation-related gaze behaviours to incorporate into our proposed framework.

A. Shared Autonomy

One strategy for mitigating the difficulties of pure teleoperation is to fuse autonomous robot planning with the user signal, a process called *shared autonomy*. Shared autonomy [2] complements users' high-level controlling abilities with the robotic system's ability to perform in order to mitigate the difficulties of completing a task. As a form of assistance, it aims to alleviate the difficulties of operating fully teleoperated systems by integrating the robot's actions seamlessly with the human actions.[2, 5] The integration of shared autonomy into robotic arms, wheelchairs, and similar systems has managed to reduce overall task time and fatigue, and more technically, minimized the cost-to-go towards a goal. [8]

Shared autonomy operates around two concepts: Goal inference and assistance. While some assistive algorithms have approached goal inference with a predict-then-assist framework, which selects an exact goal rather than general location, these systems do not continuously update for changing goals and purposes. [2] Thus, to better adapt to changing tasks, Shared Autonomy infers the user's goals as a location belief distribution which is continually updated as the user provides information through an input device. The robot then uses the belief over goals in its planning using a POMDP. [2, 8]

A number of approaches for predictions and protocols for implementing the actions that follow have been developed. In prior works from which we build on in this paper, probabilistic methods such as the use of Hidden Markov Models (HMMs) [9, 10] and Maximum Entropy Inverse optimal Control (MaxEnt IOC) systems[4, 11] have been adapted to generating probabilities. Using the controller input's location history, these models generate a probability for the goal's probable location based on the robot's movement. This probability is then used by different protocols for shared autonomy such as the Blend Method, which considers User actions and full autonomy as two independent inputs and combines them to generate assistance. [12] Another protocol, Hauser's Policy Method, creates a distribution over the goals which minimizes the burden generated by the system. [2, 8]

While these different protocols deliver assistance at some point in the task, they don't act very early on in the task when assistance is most crucial.[6] This delay happens because inferences require a large amount of information which users cannot provide at the beginning of the task using explicit inputs like the joystick. [6] As such, based on the recommendations of previous research, we are adapting more implicit means of input that are indicative of user's goal for early predictions, specifically eye gaze.[5]

B. Eye Gaze

People naturally communicate large amounts of information about their intentions through only their eye gaze [13, 14] People use gaze to obtain new information faster than any physical reaction could, especially to gather information about objects and scenes. [6] Thus, people telegraph their intentions to manipulate certain objects based on where they are looking. For example, a person making tea looks at a kettle before picking it up [15], or a person looks at a morsel on a plate before using a robotic arm to spear it with a fork. [6]

As a result of these observations in hand-eye correlated behavior, researchers we able to document a number of patterns

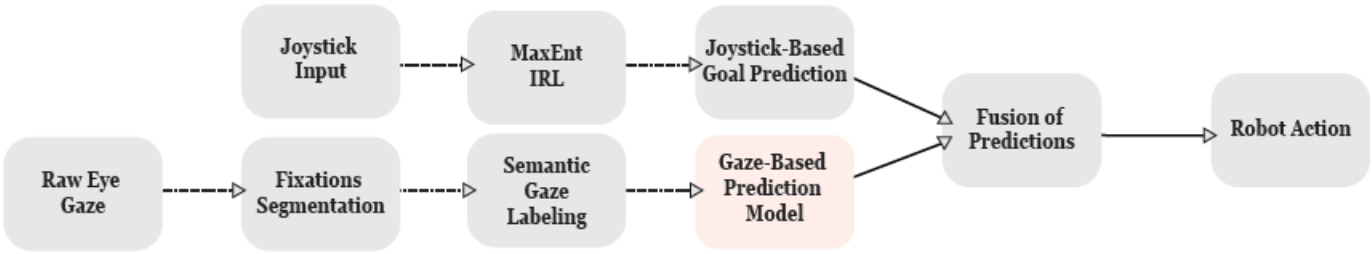


Fig. 2. Pipeline for inputs into our intended prediction system

associated with gaze and task completion. For example, Land et al noted that eye gaze reaches a target before manipulation occurs.[14] Johansson et al noted that users' gaze will fall onto all the relevant landmarks in a scene and avoid irrelevant objects to the task. [16]. Gaze was also was established to occur in a dynamic order which collects the most useful information about a scene. [13] These series of glances also illustrated that people look at their goals more than other objects in the scene and for a longer duration of time. [14, 15, 16]. While these behaviors were observed in direct hand manipulation, Aronson et al[6] were able to confirm that these patterns also occurred during robot teleoperation. Their work notes two categories of gaze during teleoperation: monitoring or planning. These monitoring and planning glances illustrate similar behaviors as by-hand manipulation.

Considering prior work that has combined robot and eye gaze [17], these finding incentives the premise that gaze could be a useful prediction source for assistive robots. [13]

III. Approach

In order to incorporate eye gaze predictions into the shared autonomy framework, we began by developing a model for eye gaze predictions based on previously documented gaze patterns. [6, 15] . After developing a number of models, we then validated them using a pre-existing dataset. Based on the results of that validation, a model will be chosen and incorporated into the shared autonomy algorithm. This will be followed by a user study to evaluate the effectiveness of our selected model.

A. Model Development

The model designed in this paper is intended to fit into the framework illustrated by Fig. 2, which shows how the two inputs of joystick and gaze will both generate a probability which will be combined for more accurate goal predictions.

While the the joystick predictions will still be generated using the current state-of-the-art systems presented by Jadvani et al [2] using MaxEnt IOCs, gaze requires a new pipeline into the system. For that, raw eye gaze will be tracked using eye-tracking glasses, then the data will be segmented into fixations. These fixations will be labeled by semantic gaze labeling using the system improved by Aronson et al [9], and finally, the labeled data will be filtered by our model to generate the probability distribution of the goal.

To develop our model, we refered to a number of distinct gaze patterns which occur during the task of food-spearing with a robotic arm.[6] These patterns which arise during manipulation provide the foundations for different models which we can develop to predict a goal from gaze. [6] Some of the documented patterns in gaze have stated that people follow their goals closely in a task, implying that users look at their targets:

1) More frequently
2) For a larger fraction of time during the task than any other item in the scene

From these implications, we developed 2 initial score tallying models which measures the user's gaze probability of falling on the goal a certain number of times or for a certain duration of time.

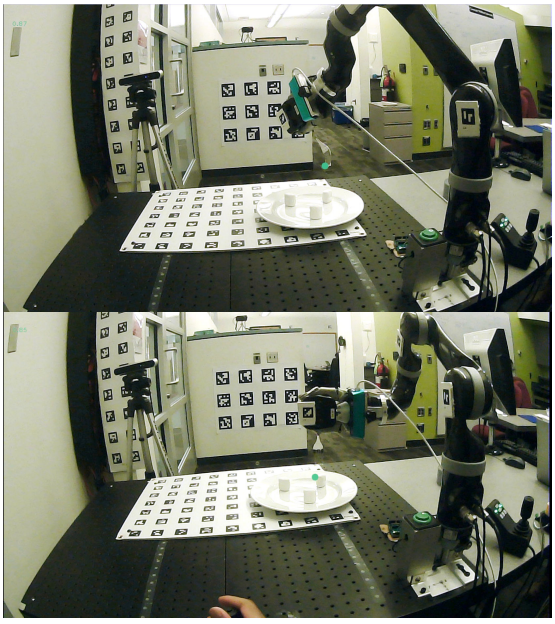


Fig. 3. Screenshots of video data from the HARMONIC dataset with gaze fixation information

B. Model Validation

After developing the different models, they were validated using the HARMONIC dataset which collected video and gaze data from users working with the Kinova robotic arm

on a food-spearing task. [18]. The experiment designed in this dataset, shown in Fig. 3, asks users to spear one of three morsels on a plate using different modes of autonomy including manual teleoperation, shared autonomy (adapted from the policy method),full autonomy, and the Blend method. This dataset segmented and filtered gaze data while also providing processed joystick input which not only allowed us to measure the accuracy rates of our models, but also allowed us to learn more about user's actions during robot teleoperation.

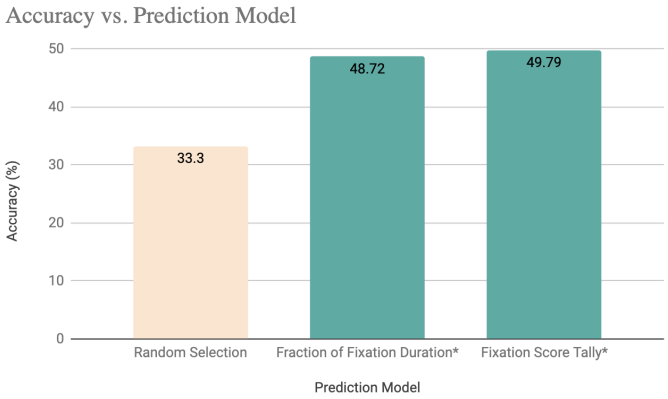


Fig. 4. Comparative analysis of prediction model accuracy between the baseline of a random selection and the models developed in this project.

We found that, as seen in Fig 4, when compared to the random selection probability in HARMONIC's three marshmallow scene, gaze-based goal predictions were about $\sim 20\%$ more likely to select the right marshmallow. While this difference is only relatively higher, it is indicative of the potential that gaze has for higher accuracies in predictions which we can utilize in future, more optimized, models.

IV. Future Works

While this work is still in progress, we hope to determine which model for gaze prediction will be most optimal for our goals. After choosing a model, we aim to conduct a user study which compares user experience and collects observations across different parameter. We hope to find that the use of gaze as a prediction method will work in favor of our goals.

A. User Study Setup

The projected set up for this project slightly augments the procedures as presented in the HARMONIC data [18]. As shown in Fig. 5, our set up adapts the use of the feeding experiment. However, it differs in the configuration of the morsels and our procedural parameters.

Instead of evaluating different types of control with the robot, we will evaluate:

1) Teleopertion
2) Shared autonomy using joystick-based predictions only
3) Shared autonomy using gaze-based predictions only
4) Shared autonomy using both gaze and joystick prediction

Fig. 5. Screenshots of video data from the HARMONIC dataset with gaze fixation information

From this experiment, we will conduct a survey evaluating users' experiences with each setting. We also intend to collect more gaze data in order to continue developing our gaze-prediction model into a more optimized format.

B. Expected Results

From this user study, we hope to conclude that the fusion of both joystick goal predictions and gaze-based goal predictions will result in an accurate and comprehensive goal probability distribution. We expect it will provide a more user friendly experience characterized by faster task completion and efficient assistance.

V. Acknowledgements

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

References

[1] Kinova, "Robotic arm series." [Online]. Available: https://www.kinovarobotics.com/en/products/robotic-arm-series

[2] 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, Jun. 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364918776060

[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)*. Christchurch, New Zealand: IEEE, Mar. 2016, pp. 35–42. [Online]. Available: http://ieeexplore.ieee.org/document/7451731/

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

[5] H. Admon and S. Srinivasa, "Predicting User Intent Through Eye Gaze for Shared Autonom," p. 6.

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

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

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

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

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

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

[12] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, Jun. 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913490324

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

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

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

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

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

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

Training an Instance Segmentation Object Classifier from Exclusively Synthetic Data

Colin Chen[1] and Christoph Mertz[2]

Abstract— **Deep-learning based object classifiers can accurately segment and classify multiple classes in a single image. However, training these models requires large amounts of pixel-wise annotations to train on. These annotations are traditionally done by hand and it takes a large amount of time to generate a sufficiently large dataset to train an instance segmentation object classifier. To address both the time and accuracy of annotating images, we introduce a method for automatically annotating images for instance segmentation. We recreate a three-dimensional model of the object and paste instances of the three-dimensional object against many different backgrounds to quickly create thousands of images with perfect pixel-wise annotations.**

I. INTRODUCTION

Recent advances in deep learning have been instrumental in creating more powerful object classifiers [1]. Convolutional Neural Networks (CNN) can single instances of objects but struggle with multiple detections in an image. Region based Convolutional Neural Networks (R-CNN) can identify multiple objects in an image by using a CNN at multiple locations on the image. Multiclass object detection can be done in real-time with Faster R-CNN [2] or YOLO [3], which take different approaches to apply a CNN at multiple locations in an image. These models can identify multiple types of objects in a single image by drawing a bounding box around each object. Multiclass instance segmentation can be done with Mask R-CNN [4]. In addition to drawing a bounding box, Mask R-CNN can create a predictive mask or outline of an object. In other words, Mask R-CNN [4] can identify which pixels belong to a certain object in addition to the object's position in the image.

All of these approaches require thousands of annotated images to train on. Datasets such as Microsoft's Common Objects in COntext (COCO) [5] provide pixel-wise annotations for multiple classes which allows models such as Mask R-CNN [4] to generate predictive outlines of an object in addition to the traditional bounding box. Training an instance segmentation classifier such as Mask R-CNN [4] requires thousands of carefully annotated images that can take humans thousands of hours to create. Each image requires careful review and the resulting annotations can still contain minor inaccuracies. While existing datasets are sufficient to create multiclass instance segmentation classifiers such as Mask R-CNN [4], it is time consuming to create new dataset for new classes.

Instead of annotating thousands of real images, another approach is to create a synthetic dataset. Synthetic images are images created partially or completely by a computer instead of captured and annotated by humans. The advantage of using synthetic data is that images can be annotated automatically. As images are being created, their class, location, and other relevant information can automatically be recorded which significantly reduces the time to create a sufficiently large dataset.

Synthetic images can be created using a variety of methods. One approach is to capture images from a realistic three-dimensional environment. This ensures objects are to scale and interact with their environment realistically. Video games are an excellent source of synthetic data especially given the improvements in computer graphics [6]. However, this type of environment is difficult to create and not always freely available.

A simpler approach is to overlay objects on a real image [7]. This approach requires cutouts of an object and background images to paste instances on. However, no virtual elements are required and it is easy to create images with different backgrounds.

We create synthetic images with a combination of these methods. We recreate a three-dimensional model of the object to preserve texture and capture two-dimensional views at any angle. We then rotate, scale, and paste these instances against multiple backgrounds to quickly create a perfectly annotated image dataset. We find that training exclusively on synthetic images can produce a model that performs well on real images.

II. RELATED WORK

The tools we use for our pipeline are structure-from-motion three-dimensional reconstruction, synthetic image generation, and instance segmemntation. In the next three subsections, we will discuss common implementations of these tools.

A. Three-Dimensional Reconstruction

Three-dimensional reconstruction is a well-researched computer vision problem [9]. There are many implementations of structure from motion and multi-view stereo algorithms that reconstruct a three-dimensional model from a series of images taken from a single camera. We use COLMAPS's [10] implementation of structure from motion and multi-view stereo algorithms to create our three-dimensional model.

[1]Colin Chen's RISS participation in the 2019 cohort was under the title "RISS Program Intern," and not as a Robotics Institute Summer Scholar. This role involved a mixture of research and administrative support. Chen is a robotics engineering student at the University of California, Santa Cruz. coachen@ucsc.edu

[2]Christoph Mertz is a project scientist at Carnegie Mellon University's NavLab.

B. Creating Synthetic Images

There are many approaches to creating synthetic data for object detection. [6] uses a video game environments to capture images from virtual environments. We use [8] to create simple synthetic images with minor adaptations to produce pixel-wise annotations in addition to the traditional bounding boxes.

C. Instance Segmentation

Deep-learning based object detection is a very popular field of study. There are many different networks for multiclass object detection such as Faster RCNN [2], YOLO [3], and Mask RCNN [4]. We modify Mask RCNN [4] to segment and identify basketballs but make no changes to the network structure.

III. METHODS

A. Structure-From-Motion Three-Dimensional Reconstruction

To recreate a three-dimensional model, we used COLMAP's [10] implementation of structure-from-motion and multiview-stereo algorithms. These algorithms take a set of images from different angles, find matching features, and create a point cloud and three-dimensional model.

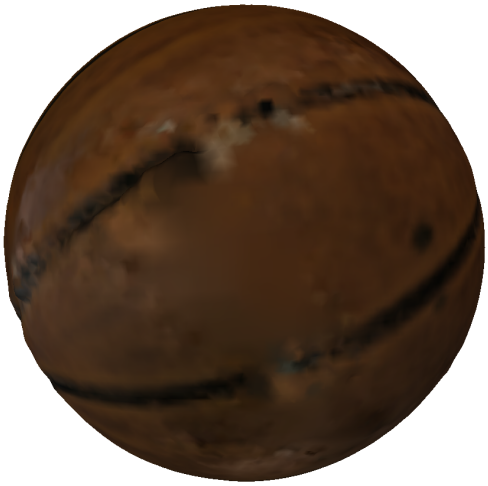
We tried several techniques to capture a comprehensive set of images from all angles. We initially placed the object on a narrow pole and took pictures by moving around the object. We then removed the pole from the three-dimensional model by hand but found that the missing area occluded by the pole produced a rough model with minor craters and inaccurate textures (Fig. 1).

This model (Fig. 1) was likely already good enough for our purposes, but we found we could get an even better model by taking pictures against a featureless background (Fig. 2a) and rotating the object to capture it from all angles (Fig. 2b) produced an even more complete model without any craters or missing textures.

Taking pictures against a featureless background (Fig. 2) significantly decreased the number of extraneous background points that needed to be removed later.

We used MeshLab [11] to remove extraneous points from the reconstructed point cloud (Fig. 3) and create a Poisson blended mesh which we used to create object instances for dataset synthesis.

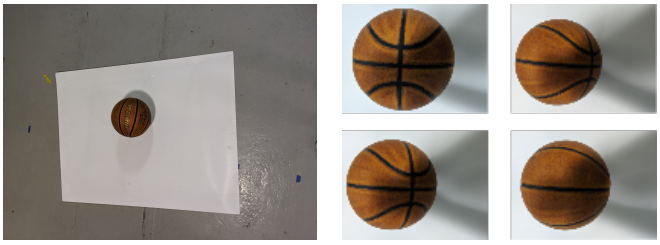
This model (Fig. 4) is of excellent quality for creating synthetic images for our classifier. However, it may not be good enough for other applications. For example, if we wanted to create a classifier that identified this particular basketball from other basketballs, we would need a more precise model. If we were to pursue a higher quality model, we could take several approaches. The main problem with the current three-dimensional model is its inconsistent coloring. This is a result of different lighting conditions on the same feature during the image capturing stage. We could create a better environment for collecting images with controlled lighting from multiple angles. We could also solve this problem in MeshLab [12] by applying more advanced coloring techniques such as texture mapping. However, the current three-dimensional model is more than sufficient for our purposes.



(a) Reconstructed Poisson Mesh blurry texture.

(b) Reconstructed Poisson Mesh with inaccurate shape.

Fig. 1: Initial Poisson blended mesh



(a) Basketball against a featureless background

(b) Image instances

Fig. 2: Input images to reconstruct our three-dimensional model

B. Creating a Synthetic Image Dataset

To create an image dataset, we modified [8] to create pixel-wise annotations for each image. [8] takes instances, rotates and scales them, pastes them against a background, and performs a variety of blurring techniques to create robust training data. We used CloudCompare's [13] animation plugin to create 2100 instances (Fig. 5b) from the three-dimensional model. We initially used 20 background images (Fig 5a a), but eventually used 40 to improve the final model's performance. Using [8], we were able to easily create 4300 images with perfect pixel-wise annotations. This

Fig. 3: The resulting point cloud from COLMAP[10].

Fig. 4: The final Poisson blended mesh model.

(a) Background images (b) Image instances

number is comparable to the number of annotated images for a single class in the COCO dataset [5].

C. Training

We trained on top of a pre-trained classifier [14] for the objects in the COCO dataset [5]. We trained with 4300 synthetic images which were created from 2100 basketball instances pasted onto 40 backgrounds. We validated on a set of 3300 synthetic images created from the same 2100 basketball instances pasted onto 5 new backgrounds. We trained for eight hours on a GeForce GTX 1070/PCIe/SSE2 but found the model began to overfit after four hours.

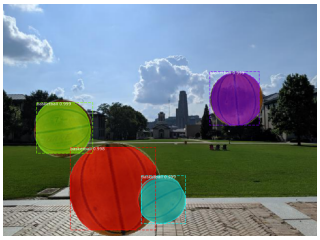
IV. RESULTS

To evaluate our model, we tested it on both synthetic images and real images. We calculated the average precision for Intersection Over Union (IOU) values from .5-.95 with .05 increments. We calculated IOU with predicted bounding boxes and ground truth bounding boxes.

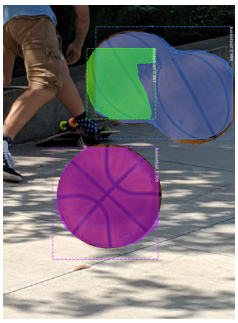
A. Synthetic Images

We evaluated our model on 500 synthetic images. These images were created with the same 2100 view instances pasted on five new backgrounds. We achieved an average precision of .82. As seen in Fig. 7, our model performs with high precision at all recall values for most IOU values. Our model has difficulty at high IOU values where a prediction must very closely match the ground truth to count as a correct detection.

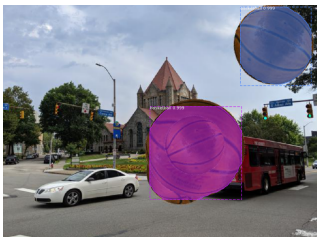
Most missed detections were due to occlusions 6c. False positives 6d were fairly rare but did occur occasionally.



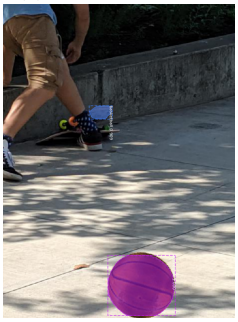
(a) Correct detections



(b) Inaccurate masks



(c) Two missed detections due to occlusions



(d) One false positive

Fig. 6: Predictions on synthetic images

B. Real Images

We also evaluated our model on 70 real images. We used [15] to hand annotate these images. We achieved an average precision of .69. As seen in Fig. 9, our model performs with high precision at most recall values over almost all IOU

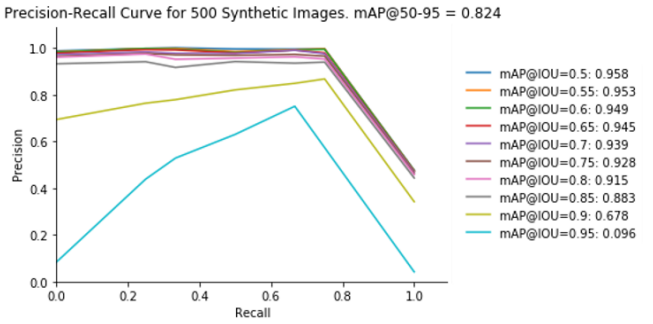


Fig. 7: Our model performs with high precision on synthetic images. High precision indicates our model correctly identifies objects that are actually basketballs. Higher IOU values mean the prediction must more closely match the ground truth to count as a correct prediction.

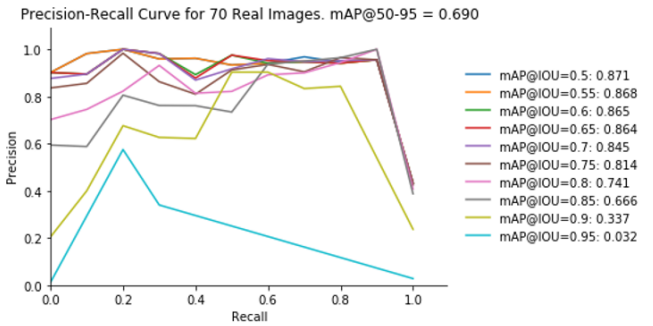


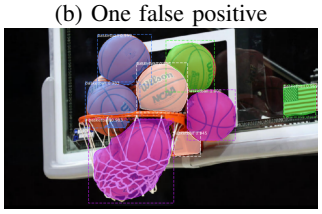
Fig. 9: Our model performs moderately well with high precision on real images. High precision indicates our model correctly identifies objects that are actually basketballs. Higher IOU values mean the prediction must more closely match the ground truth to count as a correct prediction.

values. Similarly to synthetic images (Fig. 7), the model struggles on high IOU values.

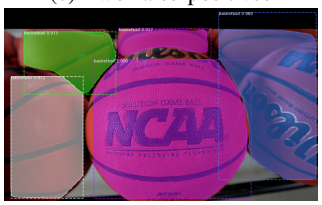
(a) Correct detection

(b) One false positive

(c) Two false positives



(d) Misses and false positives



(e) Misses due to occlusion

(f) Missed detection due to occlusion

Fig. 8: Predictions on real images

We found that across both synthetic and real images, our model rarely missed instances. Many failed detections were due to occlusions by other basketballs (Fig. 6c). We found false positives were rare in synthetic images (Fig. 6d) but fairly common in real images (Fig. 8b, 8c, 8d).

These numbers compare to the pretrained model we trained on top of. The pretrained model achieved a mean average precision of .50 across all 80 classes in the COCO dataset and .48 for the sports ball class in the COCO dataset [14].

We wanted to compare our model with the Mask R-CNN [14] trained on hand-annotated images from the COCO [5] dataset. The closest comparison is the sports ball class found in the COCO [5] dataset. The pretrained model [14] was trained to recognize 80 classes while we retrained our model to only recognize basketballs. Additionally, the sports ball class in the COCO dataset contains many types of sports balls, not just basketballs [5]. However, the pretrained model does provides a basic baseline for our models performance.

V. CONCLUSION

We could improve the precision of our model by increasing the diversity of our background images. After our initial training and reviewing the model's performance, we added more background images and saw its performance improve. To improve the accuracy of generated masks, we can add additional distractor objects. The only times the training dataset saw occlusions were when a basketball was occluded by another basketball, and as a result, any generated masks have rounded edges, even when straight edges are more appropriate.

We believe this pipeline is a good way to generate synthetic data for image detection and instance segmentation for certain objects. The next steps in developing this pipeline are to use more complex shapes with less texture and more complex geometry.

ACKNOWLEDGEMENTS

Colin would like to thank Rachel Burcin, John Dolan, and the RISS program for the opportunity to work on this project.

REFERENCES

[1] U. Shah and A. Harpale, A Review of Deep Learning Models for Computer Vision, 2018 IEEE Punecon, 2018.
[2] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1148, 2017.
[3] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection. In CVPR, 2016.
[4] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1-1, 2018.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick, Microsoft COCO: Common Objects in Context, Computer Vision ECCV 2014 Lecture Notes in Computer Science, pp. 740755, 2014.

[6] Mark Martinez, Chawin Sitawarin, Kevin Finch, Lennart Meincke, Alex Yablonski, and Alain Kornhauser. Beyond grand theft auto v for training, testing and enhancing deep learning in self driving cars. arXiv preprint arXiv:1712.01397, 2017.

[7] C. J. A. Rozantsev, V. Lepetit, and P. Fua, On rendering synthetic images for training an object detector, Computer Vision and Image Understanding, vol. 137, pp. 2437, 2015.

[8] D. Dwibedi, I. Misra, and M. Hebert, Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection, 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[9] O. zyeil, V. Voroninski, R. Basri and A. Singer, "A survey of structure from motion.", Acta Numerica, vol. 26, pp. 305-364, 2017. Science, 1989.

[10] J. Schoenberger, COLMAP. UNC Chapel Hill: ETH Zurich and UNC Chapel Hill, 2018.

[11] M. Kazhdan and H. Hoppe, Screened poisson surface reconstruction, ACM Transactions on Graphics, vol. 32, no. 3, pp. 113, Jan. 2013.

[12] G. Ranzuglia, M. Callieri, M. Dellepiane, P. Cignoni, R. Scopigno, MeshLab as a complete tool for the integration of photos and color with high resolution 3D geometry data, CAA 2012 Conference Proceedings, page 406-416, 2013

[13] R. Wicks and D. Girardeau-Montaut, CloudCompare Animation Plugin. 2G Robotics Inc., 2015.

[14] W. Abdulla, Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. Matterport, 2017.

[15] Abhishek Dutta and Andrew Zisserman. 2019. The VIA Annotation Software for Images, Audio and Video. In Proceedings of the 27th ACM International Conference on Multimedia (MM 19), October 2125, 2019, Nice, France. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1148/3343031.3350535.

Towards Hierarchical Problem Solving via Natural Language Instructions

Valerie Chen[1], Kenneth Marino[2] and Abhinav Gupta[2]

***Abstract*— To successfully solve real-world tasks, robotic agents must be able to exploit contextual and structural information in the world and apply prior knowledge of relevant tasks to a new task. Complex multi-task learning is a natural skill for humans that has proven to be difficult to build into robotic systems using deep learning techniques because of a need for a large number of examples or samples of environmental interaction to learn from. Recent approaches suggest leveraging the contextual information embedded in natural language to guide the learning process. Existing work for such hierarchical tasks assume templated task structures or provide synthetic languages as guiding information, which are typically handcrafted and do not generalize to new tasks. Thus, we present a dataset which incorporates step-by-step natural language and action execution to facilitate hierarchical task learning. The dataset consists of 20k pairs of action-instruction pairs from goal-based task completion scenarios in a crafting-based world collected from Amazon Mechanical Turk. Additionally, we present a new OpenAI Gym training environment and a new method to employ the dataset in more sample-efficient learning. In future work, we will evaluate our proposed method against existing methods to demonstrate improved generalization capabilities to unseen, yet similar tasks.**

I. Introduction

From a young age, children are able to learn from instructions and demonstrations through imitation and from exploring their environments through trial and error. These two inherently human capabilities are critical to learning how to quickly adapt to new environments and acquire new skills. In machine learning research, two dominant paradigms, imitation learning (IL) and reinforcement learning (RL), respectively aim to capture these two skills to enable agents to solve tasks in real-world environments.

IL relies on access to an expert or oracle to guide the agent to learn a policy that will mimic the expert or oracle [1]. Learning from demonstrations parallels this notion where children typically learn via step-by-step instruction and demonstration. The difficulty that IL poses is the need for a large amount of demonstrations to sufficiently capture expert behavior in many possible situations.

On the other hand, RL algorithms enable agents to learn correct behaviors in complex environments by learning from making mistakes and from receiving rewards. However, environments and tasks that are characterized by sparse or delayed rewards are difficult for existing RL algorithms to solve through purely learning from experience.

To improve generalization capabilities and sample efficiency, researchers have suggested that both IL and RL can leverage the use of natural language to provide additional information that is encoded in language particularly for complex, hierarchical tasks [2]. In some tasks, language is a necessary component to solving the task. In other tasks, language can be utilized to facilitate learning. We consider the second type where language is used to facilitate learning as a means of representing and relating policies.

We make the following contributions towards this direction:

- We present a new dataset with natural language annotated sub-tasks and corresponding game-play in a crafting-based environment with over 20,000 traces on 20 crafts.
- A method to leverage the dataset to improve sample efficiency of learning complex hierarchical tasks using a hybrid of IL and RL.

II. Related Work

Existing work presents frameworks for tackling the long-term learning problem of complex tasks through structural guiding knowledge. However, these work often reduce certain aspects of the problem, including utilizing only synthetic language or assuming prior knowledge of sub-policies within the task by design. [3] provides a guiding data structure known as a policy sketch to inform the hierarchical controller of the sub-tasks within the larger task. However, assuming knowledge of an existing template for a given task limits the ability to generalize to new tasks where this template is not know.

The language component of the task can also be templated or synthetically generated. This makes the structure of the language much easier to predict compared to natural spoken or written language. [4] introduces a Stochastic Temporal Grammar to enable interpretable multi-task RL in the Minecraft environment. Similarly, the BabyAI platform [5] presents a synthetic language which models commands inside a grid-based environment. They utilize a curriculum training to approach learning complex skills and demonstrate through experimentation in their environment that existing approaches of pure IL or pure RL are extremely sample inefficient.

Others have utilized natural language for other tasks, including [6] and [7] but not focused on the multi-task learning setting. More recently, [8] demonstrates the use

[1]Valerie Chen is a student in the Department of Computer Science at Yale University, New Haven, CT v.chen@yale.edu

[2]Kenneth Marino & Abhinav Gupta are with the Visual Robot Learning Lab, Robotics Institute, Carnegie Mellon University, PA kdmarino, gabhinav@andrew.cmu.edu

of word embeddings to inform robotic motor control as evidence of particular promise for exploiting the relationship between language and control.

A recent paper [9] presents a hybrid hierarchical reinforcement learning and imitation learning algorithm for the game Montezuma's revenge by leveraging IL for the high level controller and RL for the low level controller demonstrating the potential for combining IL and RL to achieve the benefits of both algorithms. By learning meta-actions, the agent is able to learn to solve the complex game. However, their meta-actions were also hand specified. Another recent paper [10] attempts to leverage natural language instructions in hierarchical decision making focusing only on a behavioral cloning based IL algorithm. They show that the compositional structure of language can be effective for action representation.

III. Mazebase Platform

To address the limitations of existing work, we built an interface to collect human-annotated data to guide the learning model. The utility for the data is two-fold: the first is to provide an expert human policy for the task and the second is to provide automatically annotated subpolicies. The environment is one where an agent has navigation and crafting capabilities in a grid-based world. This crafting game that our interface features is adapted from an environment originally developed by [11] and is openly available on https://github.com/facebook/MazeBase. We extended the agent's capabilities from the original repository to include the crafting capabilities. We have also created a new environment reflecting this setup in OpenAI Gym to facilitate training.

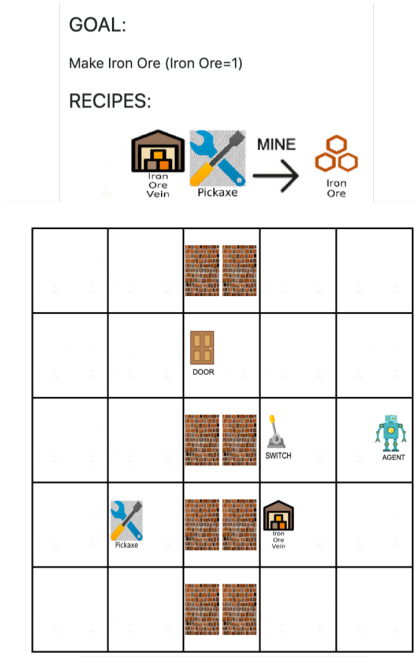


Fig. 1. Example board and goal configuration where the goal is to make an iron ore.

A. The World

As shown in Figure 1, the environment is a 5-by-5 grid-based world where there is a Crafting Agent who can navigate the grid by moving up, down, left, and right. The agent can also grab certain objects, like tools, if it is next to it and then use the tools to mine resources. The agent can also use a key or switch to open doors blocking its path from collecting other items. Finally, the agent can also go to a crafting table to build final items.

B. The Crafting Task

The tasks selected for training were scraped from the Minecraft recipes website and adapted for our Mazebase environment. In general to craft an item, the agent must first pick up a tool, go to a resource, mine the resource, and then go to a table to craft the item. There are crafts that require multiple items to be crafted first as materials for the final craft. We limited the tasks to a maximum length of 5 recipes recursively to limit the amount of time a worker would have to spend on the task. Existing work also only deals with recipe lengths of at most 2 and have already demonstrated the difficulty to learn in RL. The recipe in Figure 1 for making an Iron Ore is shown above the grid. To breakdown this recipe: the agent must use the pickaxe at the Iron Ore Vein to mine Iron Ore to complete the task.

TABLE I
List of recipes for which we are collecting annotations for labeled by the number of steps needed to complete it (depth) and other recipes which share some overlap.

ID	Recipe Name	Depth	Related Crafts by ID
1	Gold Ore	1	2, 17
2	Iron Ore	1	1, 8, 18, 20
3	Diamond Boots	2	12, 14, 16, 18
4	Brick Stairs	2	5, 7, 19
5	Cobblestone Stairs	2	4, 7, 13, 15, 19
6	Wooden Door	3	7
7	Wood Stairs	3	4, 5, 6
8	Iron Ingot	3	2, 20
9	Leather Leggins	3	10, 11, 12
10	Leather Chestplate	3	9, 11, 12
11	Leather Helmet	3	9, 10, 12
12	Leather Boots	3	3, 9, 10, 11
13	Stone Pickaxe	5	5, 14, 15, 19
14	Diamond Pickaxe	5	3, 13, 16
15	Stone Shovel	5	16, 19
16	Diamond Shovel	5	3, 14, 15
17	Gold Boots	4	1, 18
18	Iron Boots	4	2, 3, 8, 20
19	Stone Stairs	4	4, 5, 7, 13, 15
20	Iron Door	4	2, 3, 8, 18

As seen in Table I, many crafts share similar features and sub-steps that are required to finish that craft. This is done intentionally, so that later we will be able to test whether the method is able to learn these shared features and reuse existing knowledge to solve new, but related tasks more quickly.

C. Data Collection

We now describe our dataset, the collection interface and process on Amazon Mechanical Turk (AMT), and analyze

the dataset. The concrete task for our Mazebase environment is the following - given a goal craft, relevant recipes to reach the goal, and the initial board state, the worker should provide step-by-step instructions accompanied with execution on the actual game board of each instruction. So the workflow would be to type one instruction, execute the instruction, then type the next instruction, and execute until the goal was completed. We motivate this type of annotation in the demo for our task by telling the worker to imagine that they were describing how to solve this board game in a step-by-step fashion to their younger sibling.

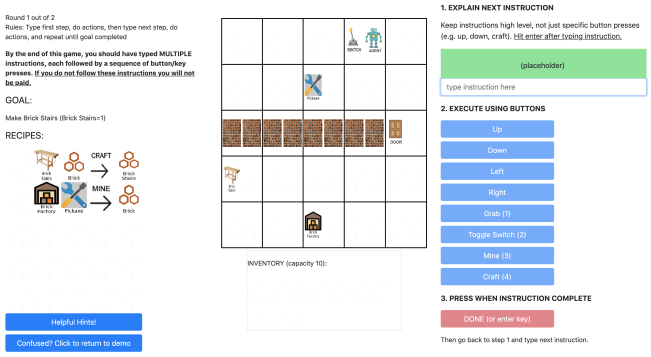


Fig. 2. Example view of game interface that the worker would see on AMT. On the left the worker is given the goal and recipes; the board is in the middle; the worker provides annotations on the right.

Consider the example in Figure 2. The agent must first go to the switch and toggle the switch to open the door. Or the agent can first choose to pick up the pickaxe. Either way, the door must be opened for the agent to complete the task, which will include mining the ore and then crafting. We expect each of the previously described high-level instructions to be a separate step that given by the user and followed by the appropriate control of the agent to execute such instructions. This type of granularity of specification was difficult to convey to the worker without explicitly giving them an example of what an instruction was. We deliberately chose not to provide examples as to not prime the worker to a particular format to follow. Originally, some workers provided not enough instructions, meaning that they wanted to finish the task as quickly as possible, and other workers provided instructions that were too granular, meaning that they did not abstract the task into sub-tasks and rather wrote “press left” or “go up 1” as their instruction. In either case, we checked prior to the worker submitting the HIT and built in precautions so that they had to redo a level if they did not comply with such instructions clearly delineated in the demo. New workers were given two short games to complete to familiarize themselves with the environment. Returning workers were given one longer game to complete as they already had experience with the task. Workers who completed the task as previously described were fully compensated.

On AMT, we were not able to make use of existing templates that were already provided, so we redirected workers to our own website where the game was hosted. We built a website from scratch using HTML, CSS, and Javascript, with the Python Mazebase game in the backend. Workers were provided with an entrance code at the beginning of the task to enter the website and an exit code when they completed the task to be able to submit the HIT. This enforces that we do not have workers doing extra HITs that we are unable to pay for and to ensure that worker who submit HITs have indeed completed our task. Then we also wrote a parsing script to be able to quickly verify all submitted HITs before payment. Our entire data-collection infrastructure and modified Mazebase environment is publicly available at https://github.com/valeriechen/dialog-rl.

D. Dataset Analysis

We present some high-level analyses of the data that we have collected thus far. Overall, as shown in Figure 3, we had many returning workers to our task. A few even wrote emails to our requesting email to let us know that they really enjoyed the HIT and thought it was quite interesting to work on. We gave bonuses to the top workers who completed the most HITs and found that over time, they were able to learn all of the built-in keyboard short cuts to complete the task quicker.

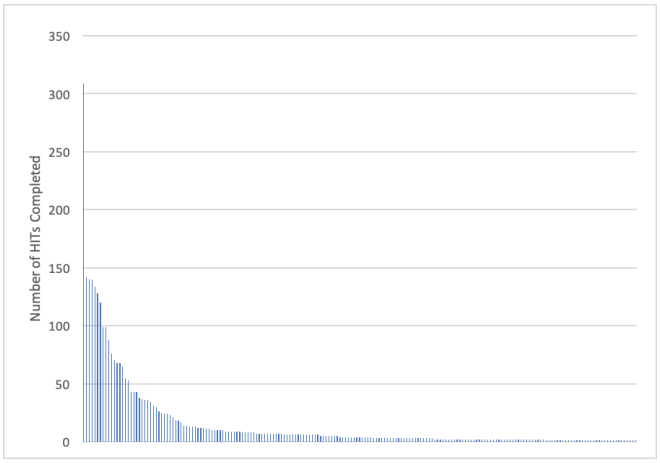


Fig. 3. Number of HITs sorted by the workers who have done the largest number with a maximum of 309.

We also provide an example of a game trace in the transcription below where the crafting task is to make Leather Boots:

Turkerdow

: Grab the key : Open the door : Grab the sword : Mine the rabbit to get rabbit hide : Craft the rabbit hide into leather : Craft the leather into leather boots

In the end, we believe that we found a happy medium in terms of the number of instructions the workers provided compared to the number of actions through the correction mechanisms that we implemented.

IV. Methods

A. State Representation

An important consideration for methods is how to best represent the state information for learning. We choose to

embed the goal information using the Glove word embedding [12] as well as each object name on the 5-by-5 grid into word vectors. We hope that this contextual information will eventually help with generalization to similar tasks that utilize different materials in its construction.

B. Reinforcement Learning Baseline

To define the task in RL, we choose a sparse reward, where the agent only receives a reward when it has completed the full craft. The magnitude of the reward can also be scaled by the number of steps the agent took to reach the final reward.

While we want to learn,

$$\theta^* =_\theta E_{\tau \sim p_\theta(\tau)}[\sum_t r(s_t, a_t)]$$

where θ^* is the optimal parameters, τ is a sampled trajectory, θ a set of parameters, s_t the state at time t, and a_t the action from time t. The agent only receives a reward when reaching goal state:

$$r(\tau) = \begin{cases} 1 & goalstate \\ 0 & otherwise \end{cases}$$

As the RL baseline, we used a standard implementation of the proximal policy optimization (PPO) algorithm [13] with the reward defined as above and the environment previously described.

C. Imitation Learning Baseline

The first IL baseline we consider is the most naive approach of behavioral cloning, where we learn a policy to determine the action given state and goal information with a MLP set-up where the state and goal are separately passed through fully-connected layers and then concatenated downstream. We will also consider the generative adversarial imitation learning (GAIL) method [14], where we make use of the sparse-reward function as defined from above and the labeled expert trajectories collected in our dataset in the game-theoretic framework.

D. Proposed Method

The proposed method consists of 3 components, the first two of which are supervised the dataset and the third which utilizes the defined reward function.

1) Language Generation: We would like to first pre-train an LSTM to generate proposed natural language sub-task instructions given the current board state and overall goal, using all annotations of step-by-step instructions that we have collected as the text corpus.

2) Imitation Learning Component: Then we proceed to utilize the expert trajectories in our dataset in a supervised manner to train a policy conditioned on the language sub-step and current state to predict corresponding action trajectories. At the time of inference, we use the pretrained language generation module to provide the language sub-step to predict actions from.

3) Reinforcement Learning Component: Finally if supervised learning was not sufficient to learn the task then we use the sparse-reward to fine-tune the model using RL.

V. Results

A. Baseline Comparison

We want to compare a baseline against pure RL, using the proximal policy optimization (PPO) algorithm, to do a 1-step craft task. Based on Figure 4, learning the 1-step task requires over 10^6 timesteps where the agent receives a reward 5 when the task is complete and 0 if incomplete.

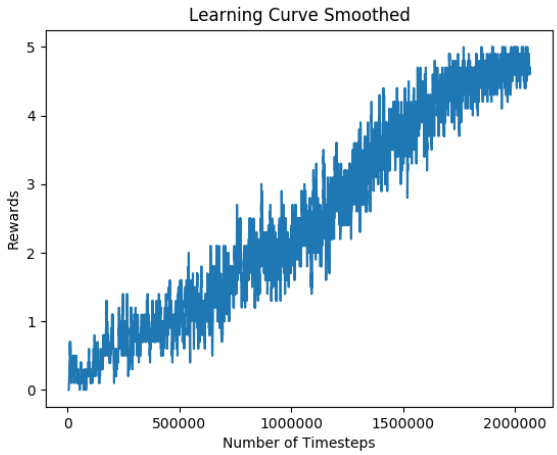


Fig. 4. Reward graph for PPO solving a simple 1-step task.

VI. Conclusion and Future Work

The future work includes the complete implementation of the described baseline algorithms and proposed methods. We would like to demonstrate generalization capabilities to unseen tasks through our use of word embeddings in the state representation and to demonstrate better sample efficiency when compared to the baseline methods given only a limited number of demonstrations. In conclusion, this project introduces a new dataset to facilitate the use of natural language in guiding learning policies for multi-step complex tasks through our proposed method using a combination of IL and RL.

VII. Acknowledgements

This work was supported by the National Science Foundation through the Robotics Institute Summer Scholar program. Thanks to Rachel Burcin and John Dolan for coordinating the RISS program and to those in the Visual Robot Learning Lab, particularly Kenny Marino and Prof. Abhinav Gupta for their guidance through the project.

References

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 21, 2017.

[2] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A survey of reinforcement learning informed by natural language," *arXiv preprint arXiv:1906.03926*, 2019.

[3] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 166–175.

[4] T. Shu, C. Xiong, and R. Socher, "Hierarchical and interpretable skill acquisition in multi-task reinforcement learning," *arXiv preprint arXiv:1712.07294*, 2017.
[5] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, "BabyAI: First steps towards grounded language learning with a human in the loop," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJeXCo0cYX
[6] E. C. Williams, N. Gopalan, M. Rhee, and S. Tellex, "Learning to parse natural language to grounded reward functions with weak supervision," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.
[7] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. DeNero, P. Abbeel, and S. Levine, "Guiding policies with language via meta-learning," *arXiv preprint arXiv:1811.07882*, 2018.
[8] D. Matthews, S. Kriegman, C. Cappelle, and J. Bongard, "Word2vec to behavior: morphology facilitates the grounding of language in machines," 2019.
[9] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé III, "Hierarchical imitation and reinforcement learning," *arXiv preprint arXiv:1803.00590*, 2018.
[10] H. Hu, D. Yarats, Q. Gong, Y. Tian, and M. Lewis, "Hierarchical decision making by generating and following natural language instructions," *arXiv preprint arXiv:1906.00744*, 2019.
[11] S. Sukhbaatar, A. Szlam, G. Synnaeve, S. Chintala, and R. Fergus, "Mazebase: A sandbox for learning from games," *arXiv preprint arXiv:1511.07401*, 2015.
[12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[14] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

Force-controlled Surface Exploration of Ultrasound Probe and 3-D Vein Reconstruction

Haoran Cheng[1] and Nicolas Mateo Zevallos-Roberts[2]

***Abstract*—Ultrasound imaging is used extensively in various clinical diagnostic and surgical procedures. However, forces exerted on the probe and probe orientation need to be fine-tuned to obtain good image qualities. In addition, sometimes the physician is required to exert significant force in uncomfortable positions during the process, which results in musculoskeletal damage. To address these issues, various robotic ultrasound systems have been developed in recent years. However, previous studies have not achieved fully automated force-based orientation adjustment of an ultrasound probe on a curved surface, which is essential for imaging quality and probe exploration. In this study, a force-based surface exploration control is proposed and preliminary studies are conducted on the UR5e robot.**

***Index Terms*—admittance control, hybrid force/posititon controlled exploration, registration with known correspondence, ultrasound image segmentation and reconstruction**

I. Introduction

During the past decade, various robotic ultrasound systems have been developed with different focus of functionalities. However, full automated control over the ultrasound probe orientation and surface exploration have not been achieved.

In some studies, the scanned surface is flat and horizontal. As a result, roll and yaw orientation adjustment of the probe are not considered. Coordinate frame definition is illustrated in Fig. 1.

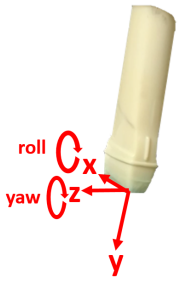


Fig. 1: Ultrasound probe frame definition

Mathiassen et al. [1] implemented compliance force control and forward flow haptic control on a tele-operated robotic ultrasound system consisting of UR5 robot and a haptic device. However the force compliance control is only implemented in the vertical direction since it is assumed to be the main contact force component.

[1] Haoran Cheng is an undergraduate mechanical engineering student at the Hong Kong Polytechnic University. haoran.cheng@connect.polyu.hk

[2] Nicolas Mateo Zevallos-Roberts is with the Biorobotics Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA nzevallo@andrew.cmu.edu

Merouche et al. [2] developed a robotic ultrasound scanner that can automatically track and reconstruct lower limb artery in a phantom. However, the phantom used in the study has a flat surface which does not always reflect the real clinical scenario.

Huang, Lan and Li [3] present a robotic ultrasound system that plans the probe path after a depth camera scan and then fine-tune probe orientation by two uni-axis force sensor attached at two ends of the probe contacting surface. However, only orientation adjustment in the vertical plane that coincide with the two force sensors can be achieved and the force sensor may block the imaging.

Other systems still rely on the sonographer's skill in adjusting the probe orientation and applying constant normal force during operation. For example, Finocchi et al. [4] developed an ultrasound system that can comply with the user's force and torque input on the probe thus following user's desired path. During imaging, the robot augments the user's applied force in normal direction thus lessening physical efforts.

Wang et al. [5] propose hybrid position and force control for surface exploration of a spherical end-effector. Due to the spherical shape of the end-effector, the normalized force vector at the tool tip can be approximated as surface normal vector and tool orientation can be adjusted accordingly. However, in the case of a rectangular-shaped ultrasound probe, This assumption does not always hold.

In order to orient the ultrasound probe locally normal to the contact surface while exploring the surface, this paper proposes a control method that implements Wang et al.'s [5] method in the case of ultrasound probe with the following objectives:

- Probe orientation adjustment in yaw direction by normalized force vector reference
- Probe orientation adjustment in roll direction by moment information
- Normal direction (y direction) admittance control to keep contact force to achieve good image quality
- Tangential velocity control to move the probe to the next target position

The fist 3 objectives are implemented and tested in the preliminary study. A 3-D vein reconstruction is conducted on a flat phantom to validate the ultrasound calibration, segmentation and reconstruction process.

II. Materials and Methods

A. System Overview

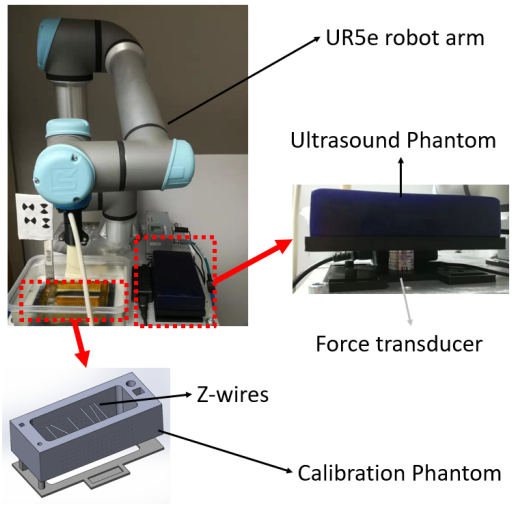


Fig. 2: System setup

Figure 2 shows the experimental system, which consists of

- A robot (UR5e, Universal Robots Inc.) holding the ultrasound probe
- An ultrasound imaging system (DIASUS Inc.)
- An ultrasound phantom (CAE Inc.)
- A calibration phantom (fCal2.0, Plus Toolkit, The Perk Lab, Queens University)
- A six-axis force/torque transducer and Net Box (Nano25, ATI Inc.)
- Custom-made 3-D printed ultrasound probe holder, calibration phantom holder and phantom plate.

Robot Operating System (ROS) framework is used to read image data, force sensor data and control the robot. The ultrasound image is captured by a frame grabber in real time and published to the */camera/image_raw* topic by *video_stream_opencv* package at 30Hz. Package *image_view* provides a service call to trigger the node *image_saver* to save images as .jpg files. The force transducer and robot communicate to the computer via Ethernet network.

B. Admittance Control of Ultrasound Probe

In the experiment, in order to perform tasks such as probe registration and placing the probe at the starting pose, it is essential for the ultrasound probe to be manipulated intuitively and interactively by the user. Though the teaching mode of UR5e robot allows the user to change robot configuration by hand manipulation, it has high torque threshold for base and shoulder joints which hinder fine adjustments. Thus, the following admittance control is implemented based on the work of Mathiassen et al. [6] and Finocchi et al. [7]

The admittance control law translates force and torque (wrench) inputs detected at the end-effector into velocity and angular velocity (twist) commands to the end-effector.

Firstly, the 6x1 desired twist in force sensor (inside the robot, between the end link and second-to-last link) frame $\mathcal{V}_f = [v_f, \omega_f]^T$ is calculated using the formula

$$\mathcal{V}_f[i] = \alpha \mathbf{K} \mathcal{F}_f[i] + (1-\alpha)\mathcal{V}_f[i-1] \tag{1}$$

where α is the constant of low-pass filter, $\mathbf{K}$ is the diagonal matrix of scaling factors and $\mathcal{F}_f = [F_f, M_f]^\intercal$ is the wrench reading from the UR5 force sensor.

If $\|\mathcal{V}_f\| < \mathcal{V}_{min}$ the robot skips the following calculations and remains still for this time instant.

Then, the desired twist is transformed from force sensor frame to space frame, which coincides with the fixed robot base:

$$\mathcal{V}_s = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \mathcal{V}_f \tag{2}$$

The joint velocity V_j is calculated:

$$V_j = J^{-1}\mathcal{V}_s \tag{3}$$

where J is the instantaneous Jacobian matrix gained from MoveIt. If $\max_i V_{j_i} > v_{safe}$, the robot stops for safety.

Lastly, the joint speed commands are sent to the robot via *speedj* command in *URScript* package.

C. Rigid Registration with Known Correspondence

In this paper, a revised version of Sorkine-Hornung and Rabinovich's [8] method of registration with known correspondence is used to register unknown frame coordinates and calibrate ultrasound probe. Let $S = [\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_n}]$ and $B = [\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}]$ be the matrices of coordinates of correspondence points in space frame and body frame respectively. The following steps can find R_{sb} and $\mathbf{p_{sb}}$ that minimize $\sum_{i=1}^{n} \|R_{sb}\mathbf{b_i} + \mathbf{p_{sb}} - \mathbf{s_i}\|$,

1) Compute the centroid of both point sets

$$\bar{\mathbf{s}} = \frac{\sum_{i=1}^{n} \mathbf{s_i}}{n}, \quad \bar{\mathbf{b}} = \frac{\sum_{i=1}^{n} \mathbf{b_i}}{n} \tag{4}$$

2) Compute the centered vectors

$$\mathbf{x_i} = \mathbf{s_i} - \bar{\mathbf{s}}, \quad \mathbf{y_i} = \mathbf{b_i} - \bar{\mathbf{b}} \tag{5}$$

3) Compute the 3×3 covariance matrix

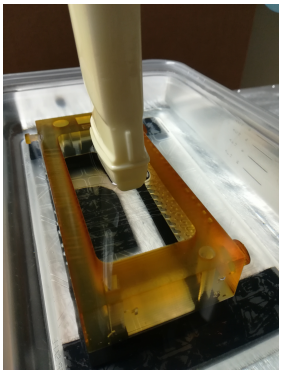
$$S = XY^\intercal \tag{6}$$

where $X = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$ and $Y = [\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n}]$

4) Compute the singular value decomposition $S = U\Sigma V^\intercal$
5) Compute R_{sb} and p_{sb} as follows:

$$R_{sb} = V \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & det(VU^\intercal) \end{pmatrix} U^\intercal \tag{7}$$

$$\mathbf{p_{sb}} = \bar{\mathbf{b}} - R_{sb}\bar{\mathbf{s}} \tag{8}$$



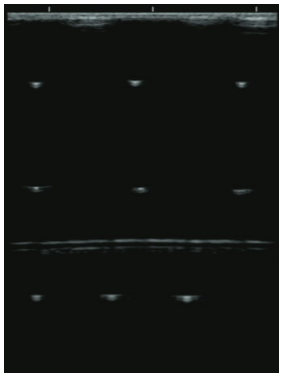


Fig. 3: (a) Calibration setup. (b) Ultrasound image

D. Ultrasound Calibration

We define the following subscripts representing coordinate frames:

- C: calibration phantom frame
- R: robot base frame
- U: ultrasound probe frame
- A: ATI force sensor frame
- I: image frame

The goal of ultrasound calibration is to find the transformation matrix T_{UI}.

As shown in Fig. 3 (a), a z-wire calibration phantom is placed in a water bath. The ultrasound probe is moved by the robot arm to several locations and the ultrasound images are recorded. Then, wire cross-sections are manually segmented in MATLAB as shown in 3 (b). Let p denote each segmentation point in image frame.

p_I and p_U are calculated as follows:

$$p_I = \begin{bmatrix} s_x u \\ s_y v \end{bmatrix} \tag{9}$$

$$p_U = T_{RU}^{-1}\, T_{RC}\, p_C \tag{10}$$

where (u, v) denotes horizontal and vertical pixels of segmentation point and s_x and s_y denote the horizontal and vertical scaling factor. Knowing the ultrasound probe and holder CAD information, ultrasound frame can be defined in the robot urdf file, thus T_{RU} can be obtained from *tf::listener*. p_C can be obtained from the CAD file of calibration phantom and the image segmentation results using similar triangle method [9]. T_{RC} can be obtained by poking the ultrasound probe in the rectangular groove on the calibration holder.

After calculating pairs of corresponding p_I and p_U, registration with known correspondence can be performed to calculate T_{UI}. The point locations after matching are shown in Figure. 4.

E. Image Segmentation

In this paper, basic computer vision techniques are used to detect vein cross-sections in ultrasound B-mode images as circles. Steps are as follows:

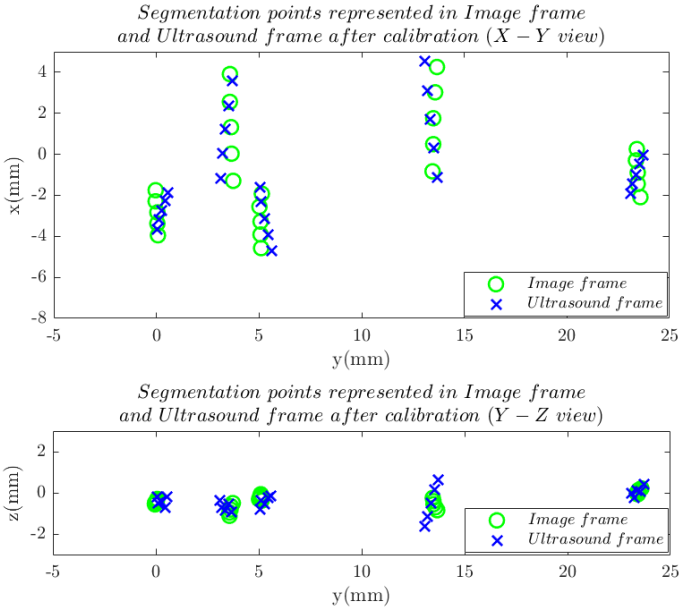


Fig. 4: Correspondence points after calibration

1) Gaussian filtering and thresholding by *ismooth()* function in *Robotics Toolbox* [10]
2) Hough transform is used to detect vein cross-sections as circles.

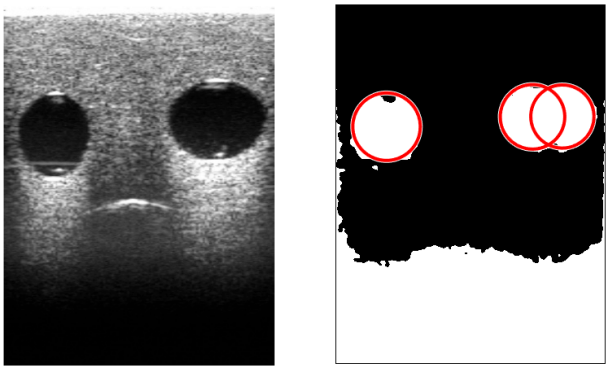


Fig. 5: (a) Before segmentation. (b) After segmentation

It is observed in Fig. 5 that ellipse vein cross-section is segmented as two circles.

F. Ultrasound Probe Force Control

The control architecture is illustrated in Fig. 6.

Contact wrench applied from the phantom to the probe in the probe frame $\{U\}$, $\mathcal{F}_U = [F_{ux}, F_{uy}, F_{uz}, M_{ux}, M_{uy}, M_{uz}]^\intercal$ is computed as follows [11]:

$$\mathcal{F}_U = [Ad_{T_{AU}}]^\intercal(-(\mathcal{F}_A - \mathcal{F}_{A0})) \tag{11}$$

where $\mathcal{F}_A = [F_A, M_A]^\intercal$ is the wrench reading of force transducer and $\mathcal{F}_{A0}$ is the force transducer reading without ultrasound probe contact.

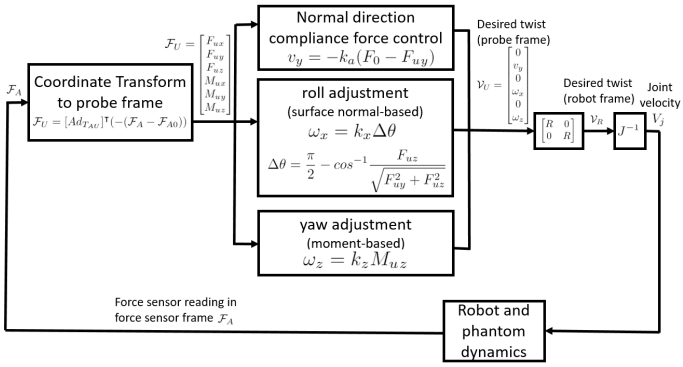


Fig. 6: Control Loop of Ultrasound Probe Control

$[Ad_T] = \begin{bmatrix} R & [p]R \\ 0 & R \end{bmatrix}$ is the adjoint representation of the transformation matrix $T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$ and $[p] = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}$ is the skew-symmetric matrix corresponding to $p = [p_1, p_2, p_3]^\intercal$ [11]. The transformation from ultrasound frame to ATI force sensor frame T_{AU} is calculated as follows:

$$T_{AU} = T_{RA}^{-1} T_{RU} \tag{12}$$

where T_{RU} is obtained from *tf::listener* in ROS. and T_{RA} is obtained from point registration of known correspondence.

Then, the desired twist in ultrasound frame $\mathcal{V}_U$ is calculated as follows:

$$\mathcal{V}_U = [0, v_y, 0, \omega_x, 0, \omega_z]^\intercal \tag{13}$$

where

$$v_y = -k_a(F_0 - F_{uy}) \tag{14}$$

$$\omega_x = k_x \Delta\theta \tag{15}$$

$$\Delta\theta = \frac{\pi}{2} - cos^{-1}\frac{F_{uz}}{\sqrt{F_{uy}^2 + F_{uz}^2}} \tag{16}$$

$$\omega_z = k_z M_{uz} \tag{17}$$

Equation (14) is the probe normal direction (y-direction) admittance control aiming at keeping a constant contact force between the probe and phantom in probe normal direction. F_0 is the desired contact force, F_{uy} is the y-direction force in ultrasound probe frame.

Equation (15) is the surface normal-based roll adjustment. $\Delta\theta$ is the angle error between current roll and the surface normal direction, which is estimated by the angle between the projected force vector in yz plane and the -y direction. The assumption is that the force vector is perpendicular to phantom surface, which is considered to be true in roll direction where point (cylindrical) contact can be assumed. $\Delta\theta$ is calculated according to Equation (16)

Equation (17) is the moment-based yaw orientation adjustment.

Then, the computed twist in ultrasound probe frame is transformed to robot frame and then transformed to joint space using Equation (2) and (3).

III. Preliminary Results

A. Compliance Force Control with roll&yaw Adjustment

In this preliminary study, the control architecture in Fig. 6 was implemented. Once the ultrasound probe was in contact with the horizontal flat phantom, the robot arm adjusted its roll and yaw while keeping normal contact force. Fig. 8 shows

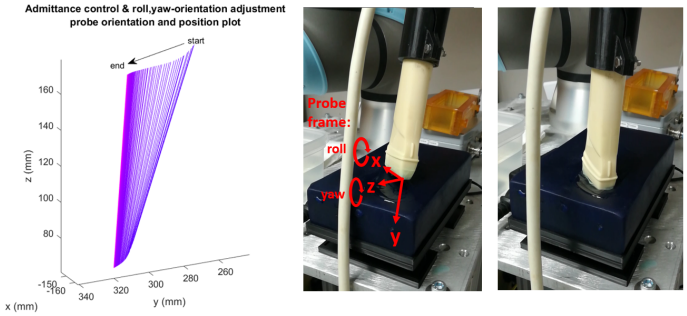


Fig. 7: (a) ultrasound orientation (b) start pose (c) end pose

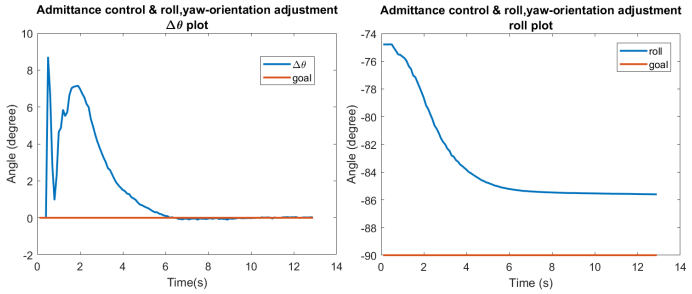


Fig. 8

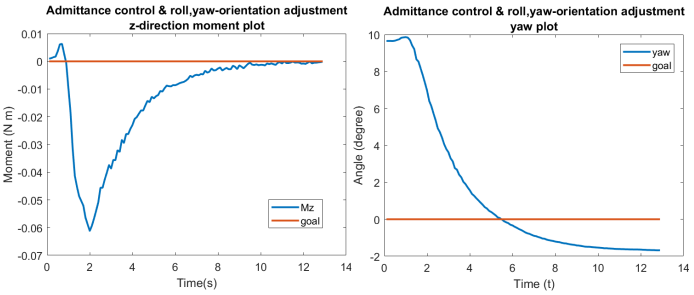


Fig. 9

the $\Delta\theta$ and roll change during the control. Figure 9. shows the moment and yaw change during the control. It is observed that $\Delta\theta$ and M_z reached the goal within 6 seconds while the orientation reached the near-vertical orientation with steady state error of 4.4 degrees and 1.7 degrees in roll and yaw.

B. Ultrasound Scan on Flat Surface and Vein Reconstruction

To validate the calibration process, a preliminary vein reconstruction experiment was conducted on a horizontal flat phantom with vertical compliance force control. The result is shown in Fig.10. It is observed that false and missing segmentation take place occasionally.

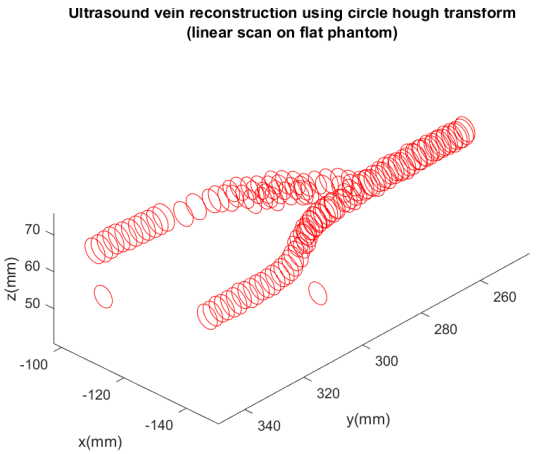


Fig. 10: 3D vein reconstructed from linear scan on flat phantom

IV. Conclusion

In this study, a force-controlled surface exploration method of ultrasound probe is proposed. Preliminary studies validate the first three control objectives and ultrasound calibration, segmentation and reconstruction.

In the future, probe exploration on curved surface and vein reconstruction can be experimented. In addition, the robustness and steady state error of the control method can be improved. Lastly, more sophisticated vein segmentation methods can be implemented.

Acknowledgment

Firstly, I would like to express gratitude to Ms. Rachel Burcin and Dr. John Dolan for organizing the Robotics Institute Summer Scholars program and the Hong Kong Polytechnic University for providing subsidy for the program.

I would like to express my great appreciation to Mr. Nicolas Mateo Zevallos-Roberts for his patient guidance. My grateful thanks also extends to Mr. Lu Li for his constructive feedback on mechanical design and plot design.

I am particularly grateful for the patient and countless assistance given by Mr. Charles Hart on ROS and hardware support. I would like to thank Dr. John Galeotti for providing the ultrasound phantom, calibration phantom and ultrasound imaging system. I would like to offer my thanks to Mr. Albert Xu for his support on software.

References

[1] K. Mathiassen, J. E. Fjellin, K. Glette, P. K. Hol, and O. J. Elle, "An ultrasound robotic system using the commercial robot ur5," *Frontiers in Robotics and AI*, vol. 3, p. 1, 2016.

[2] S. Merouche, L. Allard, E. Montagnon, G. Soulez, P. Bigras, and G. Cloutier, "A robotic ultrasound scanner for automatic vessel tracking and three-dimensional reconstruction of b-mode images," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 63, no. 1, pp. 35–46, Jan 2016.

[3] Q. Huang, J. Lan, and X. Li, "Robotic arm based automatic ultrasound scanning for three-dimensional imaging," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1173–1182, Feb 2019.

[4] R. Finocchi, F. Aalamifar, T. Fang, R. Taylor, and E. Boctor, "Co-robotic ultrasound imaging: A cooperative force control approach," *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 10135, 2017.

[5] L. Wang, Z. Chen, P. Chalasani, R. Yasin, P. Kazanzides, R. Taylor, and N. Simaan, "Force-controlled exploration for updating virtual fixture geometry in model-mediated telemanipulation," *Journal of Mechanisms and Robotics*, vol. 9, no. 2, 2017.

[6] K. Mathiassen, J. E. Fjellin, K. Glette, P. K. Hol, and O. J. Elle, "An ultrasound robotic system using the commercial robot ur5," *Frontiers in Robotics and AI*, vol. 3, p. 1, 2016.

[7] R. Finocchi, F. Aalamifar, T. Y. Fang, R. H. Taylor, and E. M. Boctor, "Co-robotic ultrasound imaging: a cooperative force control approach," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 10135, Mar. 2017, p. 1013510.

[8] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," ETH Zurich, Department of Computer Science, Tech. Rep., 2016, technical note.

[9] L. Mercier, T. Lang, F. Lindseth, and L. D. Collins, "A review of calibration techniques for freehand 3-d ultrasound systems," *Ultrasound in Medicine Biology*, vol. 31, no. 2, pp. 143 – 165, 2005.

[10] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB, Second Edition*, 2nd ed. Springer Publishing Company, Incorporated, 2017.

[11] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st ed. New York, NY, USA: Cambridge University Press, 2017.

Detection of Static Pedestrians from Vertically Sparse 3D Point Clouds

Ivana Collado[1] Luis E. Navarro-Serment[2]

***Abstract*—Precise pedestrian detection is an essential capability for autonomous mobile robots operating in populated environments. High resolution 3D LiDARs capable of producing dense point clouds, commonly used for pedestrian detection in autonomous vehicles, are economically and computationally expensive. Low resolution 3D LiDARs offer a more accessible alternative, suitable for use in small robots. However, they produce vertically-sparse point clouds, which makes the interpretation of shapes for object classification more challenging. This paper proposes a method suitable for detecting pedestrians using 3D LiDAR sensors with low vertical angular resolution. Our method handles the decreasing point density of more distant objects, and is also capable of interpreting partially-seen pedestrians at close ranges, making it suitable for operation in close proximity to pedestrians. Moreover, the approach does not rely on motion features, thus enabling the robust detection of static pedestrians. The algorithm consists of three basic steps: first, the raw point cloud data is clustered. Then, a simple screening process identifies clusters that are candidates for representing humans; a set of 3D and 2D geometrical features are obtained for each candidate cluster. Finally, the features are fed to a binary classifier trained using the Adaboost algorithm, which determines whether the cluster represents a human. We conducted experiments to compare the performance of this approach against that of the detector previously used in our robot. Experiments reveal an overall increase in performance, particularly for static pedestrian detection at distances from 1-10m.**

***Index Terms*—Sparse Point Clouds, Pedestrian Detection, LiDAR, Adaboost classifier, Perception for Autonomous Vehicles.**

I. Introduction

Autonomous robots must be capable of accurately sensing and interpreting their environment to avoid collisions. Perception of static and dynamic objects must be specially robust in scenarios where robots work in close proximity with people. To achieve safe human-robot interaction, real time detection and classification of objects is required. This makes the robot aware of potential collision situations and allows it to timely take obstacle avoidance measures.

Laser range sensors are commonly used for perception in many mobile robot scenarios. Compared to RGB camera systems, LiDARs are typically more stable against illumination changes and have larger fields of view. High resolution 3D LiDARs provide vertically and horizontally dense point clouds which are suitable for detailed detection. However, they are economically and computationally expensive. Furthermore, their larger size and power budgets makes them unsuitable for small robots. Conversely, low resolution 3D LiDARs are economically more accessible and less computationally expensive, which in turn lowers their size and power demand. However, they usually produce vertically sparse point clouds, which makes shape recognition more challenging since the point cloud density in the vertical direction is increasingly reduced as distance to detected object grows. Moreover, if the detected object is too close to the sensor, it will only be partially detected. Both long and short range detection problems using a low resolution LiDAR can be observed in Figure 1.

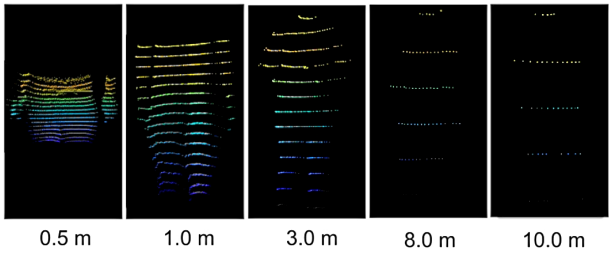


Fig. 1. Sample pedestrian point clouds obtained at various distances from the sensor using a low resolution LiDAR Velodyne VLP-16.

The proposed approach explores a methodology for close range static pedestrian detection from a vertically sparse point clouds. The focus of this research is to enhance the static human detection of a small mobile robot that uses a low resolution 3D LiDAR for 360° sensing and requires fast point cloud processing time without GPU. In our target application, the previously implemented method for human detection relied heavily on object motion and distance tracking, resulting from numerous false negatives since the static pedestrians were not detected. The proposed approach calculates a geometric feature vector for each object in the scene, without relying on motion. The feature vector is afterwards evaluated by a binary classifier.

II. Related Work

In literature there are many different approaches for pedestrian detection. Image based approaches are very common and have been successfully implemented in numerous works, as seen in [1], [2] and [3]. Others explore multi-sensor systems, where they focus on merging vision and laser data as presented in [4]. These approaches are, as mentioned before, limited to

[1]Department of Engineering and Information Technology, Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, NL 64849, México. ivanacollado@gmail.com

[2]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. lenscmu@ri.cmu.edu

the camera's field of view and are susceptible to changes in illumination conditions.

2D laser scanners have been extensively used for human detection. The work which is closest to our work is [5], which presents a set of 14 simple 2D features extracted from a single line scan, which are fed to an Adaboost binary classifier. In order to achieve a more reliable detection, this approach is expanded to work with 3D point clouds in [6] by dividing the 3D cloud into a series of 2D point clouds at various heights. In this approach, each layer in the 3D point cloud votes toward the likelihood of the cluster being a human.

Pedestrian detection using high definition 3D LiDAR is also explored in [7], here the point cloud is clustered and a potential human cluster is divided in three sections, upper half, lower left and right halves. 3D variance features are extracted from each section and then the 3D cluster is projected onto a 2D histogram on the two principal planes. The extracted geometrical features are then processed by a classifier trained using an SVM. Motion features, like moment of inertia is also calculated and used to train a second SVM classifier for a more robust detection. In [8] two more features to the previous approach are added: the slice feature which considers widths at different heights in a 3D point cloud cluster and the distribution of reflection intensities feature. Hsueh-Ling presents an approach in [9] that extracts the 3D and 2D features presented in previous works ([7] and [8]) and it tackles the unavoidable point resolution loss while distance increases by applying a distance-aware expansion approach when performing 3D-to-2D projection. All of the approaches mentioned above were implemented using a high resolution 3D LiDAR and are not tailored to work for vertically sparse point clouds. Motion is an important feature in these pedestrian classification approaches, which contradicts our objective of detecting static pedestrians.

More resent approaches for pedestrian detection in 3D point clouds like [10] and [11] successfully use deep learning neural network approaches like RCNNs ad RPNs for training a classifier. These methods require large amounts of training data, as well as graphic processing units (GPU) and vast processing resources at run time, compared to previous machine learning algorithms. In contrast to these approaches, the Adaboost algorithm requires a much smaller training set, and its lower computing budget makes it suitable for operation in real time. In [5] and [12] the Adaboost algorithm achieved high precision rates for human and object detection using 2D and 3D LiDARs.

There are two main aspects addressed by our work. First, we explore a simple 2D and 3D feature set that does not relay on motion and focuses on horizontal characteristics to reduce the effect of increasing vertical sparsity of 3D point cloud from degrading precision in human detection. Second, our human classification model is trained with the Adaboost algorithm in order to limit run time processing and provide high precision results with a small training data set, as large labeled data sets for low resolution LiDARs are not readily available.

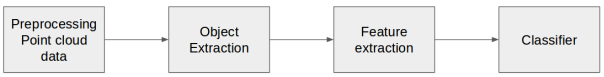


Fig. 2. General pipeline for this pedestrian detection framework.

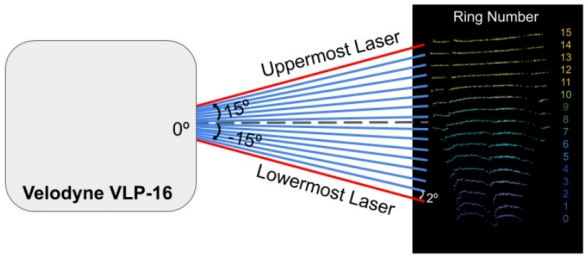


Fig. 3. Side view of the 3D sensor. The vertical field of view of $\pm 15°$ is covered by 16 independent rotating lasers, with a $2°$ angular separation between scan lines, or *rings*. Conversely, the horizontal angular resolution (i.e. angular distance between consecutive measurements in each ring) is $0.25°$.

III. Approach

In the following subsections we present the detailed methodology followed in our approach; an overview of the general data processing pipeline is illustrated in Fig.2. First, the processing flow from obtaining the raw detection data to the object detection and extraction is introduced. Then, details of the classification process, with a focus on feature extraction, are explained.

A. Preprocessing point cloud data

A 3D LiDAR typically generates range measurements arranged as a series of scan lines, called *rings*, as shown in Figure 3, where each ring is designated by a number. The point cloud $Z_j = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{N-1}\}$ read from sensor is defined as the set of N points measured at time t_j, whose elements are represented by Cartesian coordinates and their corresponding ring number:

$$\mathbf{x} = (x, y, z, r) \tag{1}$$

The raw point cloud is not down sampled, as object details are necessary for the classification process; however, only a close range region of interest, centered at the robot's current location, is considered, thus removing unnecessary background data and accelerating the process:

$$\text{Region of interest} = 20m \times 20m \times 3m \tag{2}$$

B. Object extraction

Both ground removal and clustering are necessary for detecting potential humans, in this case both steps are done in a separate code. This separate code not only down samples, but vocalizes the point cloud for faster computation. The ground is removed by creating an elevation map, the method is thoroughly described in [15]. First, a virtual 2D slice from a certain height above ground is extracted and projected onto a

TABLE I
SEGMENT FEATURES

3D Features		2D Features	
Nr	**Feature Name**	**Nr**	**Feature Name**
f_1	Number of points	f_4	Circularity
f_2	3D Covariance	f_5	Radius
f_3	Distance from LiDAR	f_6	Standard Deviation

grid plane. An elevation map is then computed by subtracting one standard deviation from the average height of all the points in the cell. Clustering is then done by using the KdTree method. Clusters require a minimum of 21 points and distances between points must be less than 0.75m. Clusters found are then considered detected objects.

We define $T_j = \{\mathbf{o}_0, \mathbf{o}_1, ..., \mathbf{o}_{I-1}\}$ as the set of I objects detected in a frame collected at time t_j, whose elements are represented by the object center using Cartesian coordinates, object volumentric dimensions and object id:

$$\mathbf{o} = (x, y, z, w, h, d, id) \tag{3}$$

T_j is then communicated to our pedestrian detection code. The objects position and size information are used to isolate, from the original point cloud Z_j, points corresponding to detected objects. We now have a collection of M sets of $\{S_0, S_1, ..., S_{M-1}\}$, where $S_{i\in\{0,1,...,M-1\}} \subset Z_j$. To be considered a pedestrian candidate, a set must contain at least 4 rings with more than 5 points. This way, only objects with the right point distribution to be potential humans are passed to the feature extraction set.

C. Feature Extraction

From each valid object S, a feature vector is extracted, which will then be evaluated by a classifier to determine if the object is human. The feature vector calculated is made up of 6 features indicated in Table I. The features are divided in two categories. Features under the 3D category are calculated using all the points in a segment, and are meant to provide a general idea of object dimensions and orientation. 2D features are calculated using only the middle third of the rings and projecting then into an XY plane. This second set of features represent the overall shape and curvature of the object.

1) Number of points: Number of points can be correlated with distance to determine if object is dense appropriate to be a human.

$$N = (\mathbf{s}_i) \tag{4}$$

2) 3D Covariance: This feature is used to determine overall orientation of segment, usually humans are in an upright position, which would prove very different from any non human object that is horizontally dominant. Only 6 values of the matrix are usually considered because it is a symmetric matrix. In this work we also do not consider σ^2_{xz} because test showed this feature had negligible influence in the classification process.

$$\mathbf{\Sigma} = \{\sigma^2_{xx}, \sigma^2_{xy}, \sigma^2_{yy}, \sigma^2_{yz}, \sigma^2_{zz}\} \tag{5}$$

3) Distance from LiDAR: As previously mentioned, this feature can be correlated with number of points in a cluster and others to determine correct dimensions of a human cluster.

$$d = \sqrt{\bar{x}^2 + \bar{y}^2} \tag{6}$$

4) Circularity: Circularity is key for differentiating humans from any flat surface, this is specially important in indoor scenarios to rule out any wall, table or desk segments that might fit the initial dimensional search. For calculating this feature the linearized geometric approach is used, explained in detail in [16]. The method involves the sum of the residual squares to a fitted circle. Given a set of points in Cartesian coordinates, it is possible to parameterize the problem by a vector of unknowns $\mathbf{z} = (x_c, y_c, x_c^2 + y_c^2 - r_c^2)^T$, where x_c, y_c and r_c denote the circle's center and radius, respectively. With this, the equation system $\mathbf{B} \cdot \mathbf{z} = \mathbf{a}$ can be established.

$$\mathbf{B} = \begin{pmatrix} -2x_0 & -2y_0 & 1 \\ 2x_1 & -2y_1 & 1 \\ \vdots & \vdots & \vdots \\ 2x_{n-1} & -2y_{n-1} & 1 \end{pmatrix} y = \begin{pmatrix} -x_0^2 - y_0^2 \\ -x_1^2 - y_1^2 \\ \vdots \\ -x_{n-1}^2 - y_{n-1}^2 \end{pmatrix}$$

The equation system can be solved using the pseudo-inverse:

$$\mathbf{z} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T \cdot \mathbf{a} \tag{7}$$

Finally, the circularity Ω is calculated from the residual sum of squares:

$$\Omega = \sum_{j=1}^{n} (r_c - \sqrt{(x_c - x_j)^2 + (y_c - y_j)^2})^2 \tag{8}$$

5) Radius: The radius r_c is obtained from the best circular fit calculated in the previous feature. The difference in radios from two clusters with similar size can help determine their shape: a flat wall sample will have a much bigger radios than a person sample.

6) Standard Deviation: The standard deviation in the XY plane is calculated as follows:

$$\sigma = \sqrt{1/(n-1)\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})} \tag{9}$$

D. Classification

As previously mentioned, we use the Adaboost algorithm to train a binary classifier. Some of the advantage of using Adaboost over deep learning approaches are: the easy implementation, high performance, low processing time and small data set required. In this paper we specifically used the Adaboost algorithm introduced by Freund and Schapire [11]. This algorithm repeatedly fits a set of weak learners, which are classifiers who's only requirement is to be better than random guessing, on a changing set of data and combining these weak classifiers through a weighted majority vote to create one strong classifier. The input of this algorithm is a labeled set of one dimensional feature vectors like the one described in the

previous subsection. The initial weights are all $w_i = 1/f$, f being the number of features used. With each iteration, examples predicted incorrectly are given more weight, while those predicted correctly are given less weight, this way, missed examples are increasingly given more importance in order to be classified correctly at the end. In this work the number of weak learners used to train a final model is 50.

IV. Experiments

The approach presented above was implemented and tested using the low density 3D LiDAR Velodyne VLP-16 which has 16 channels and a vertical resolution of approximately 2° illustrated in Fig. 3. The vertical field of view of this sensor is $\pm 15°$. The sensor was mounted on a ClearPath Husky A200 robot, as seen in Fig. 4, at a height of approximately 40cm from the ground. Various point cloud recordings were obtained from different environments, including cases where the robot was stationary and others where it was in motion. Human test subjects were recorded at various distances inside the Velodyne's field of view, the recordings also included different activities being preformed by the test subjects, including: remaining stationary, walking, running, turning and jumping.

The recorded data was used to obtain point cloud samples and for each sample set, the corresponding feature vector was calculated and manually labeled as human or non-human. These feature vectors were then used to build a data set. The final data set has a total of 1763 samples, 1214 examples are identified as stationary. The data set has a total of 415 moving human samples and 300 stationary ones.

The recorded data was compared against the motion dependent approach [18] previously used in our robot, which was used as a baseline. The previous approach segments the point cloud and identifies clusters of points representing objects and categorizes them as large (walls, cars, trees, etc.) or small objects (tables, people, chairs, etc.). This approach relies on an object tracker that determines the distance an object has moved. If an object has traveled at least a minimum distance over a span of several frames, and if it also meets other scores in terms of size and variations in size and velocity, it is considered a human. This approach, though very easy to implement and capable of running in real time on limited computing resources, was designed to focus on moving pedestrians, thus producing large numbers of false negatives when dealing with humans that are not in motion.

The objective of the experiments is to evaluate the classification results of the proposed method and compared it to the previous approach to determine which can more successfully detect static pedestrians. First, training and results of a new Adaboost classification model using only stationary segments is done implementing a five fold cross-validation technique. Individual feature importance and errors for different distance ranges are also obtained and analyzed. Next, another classification model is trained and compared, but this time, considering stationary and in-motion segments to analyse results in all scenarios. Afterwards, the model with the best results is chosen and it is validated using new never before seen stationary data obtained in a new location and recorded with different test subjects. The new dataset has a total of 382 stationary samples including 201 pedestrians.

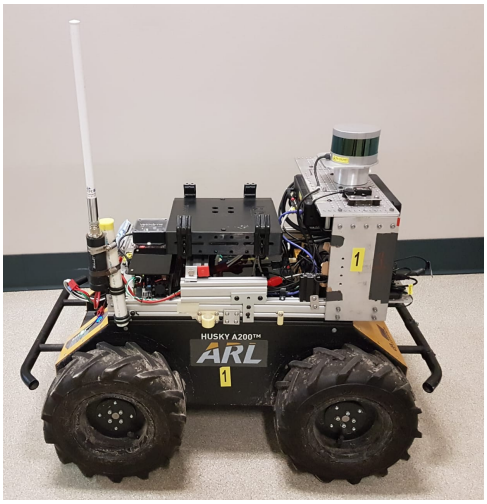


Fig. 4. Robot Clearpath Husky and sensor Velodyne 16, with which all human perception experiments were carried out.

TABLE II
PEDESTRIAN DETECTION PERFORMANCE ON TEST DATA SET

Results	Accuracy (%)	Recall (%)	F1 (%)	Precision (%)	AUC (%)
Using Static Samples					
Previous Approach	77.19	9.67	15.06	36.66	53.54
Our Approach	96.38	94.61	92.38	91.02	95.53
Using Static and Motion Samples					
Previous Approach	70.29	23.68	36.76	94.08	61.09
Our Approach	97.18	98.39	96.50	94.39	98.39

V. Results

In Table II, first we compare the performance of our human detector with that of the previously used approach [18], using 1214 static-only samples, including 300 pedestrian examples. Second, we compare using 1214 static examples, including 300 pedestrians, and 549 mobile examples, including 415 pedestrians. We observe that our approach obtains considerable better results in all metrics using only static pedestrians. When comparing the static and motion samples, we notice that our approach also preforms better; yet, the precision metric using our approach is only slightly increased. Table III shows the confusion matrix using static-only samples. The previous approach does a very good job detecting true negative samples; yet, it mislabels many human samples as non-humans (i.e. false negatives). Our approach reduces the number of false negatives by more than 80% while maintaining a high percentage of true negative

TABLE III
CONFUSION MATRIX COMPARISON ON STATIC SAMPLES

	Previous Approach Detected Label		**Our Approach** Detected Label	
True Label	Human	Non-Human	Human	Non-Human
Human	29 (10.28%)	253 (89.71%)	265 (93.97%)	17 (6.03%)
Non-Human	26 (2.78%)	906 (97.21%)	35 (3.75%)	897 (96.24%)

TABLE IV
PEDESTRIAN RECOGNITION PERFORMANCE ON NEW DATA SET

Results	**Accuracy (%)**	**Recall (%)**	**F1 (%)**	**Precision (%)**	**AUC (%)**
Using Static Samples					
Previous Approach	53.14	1.10	2.18	100.0	50.55
Our Approach	93.97	97.79	93.89	90.30	94.16

detections.

To analyze the ability of our detector to generalize beyond the examples of the training set, we studied its performance using a new data set, including only static examples never used before for training or testing. The result is shown in Table IV. The data used in this experiment has a total of 382 stationary samples including 201 pedestrians. Results of our approach with this new data were slightly lower than with the previous test; however, all metrics are acceptably accurate. The precision of the previous approach is reported as 100%. The precision metric formula is $Precision = tp/(tp + fp)$ and in this case Table V shows there were no false positives. However, due to the large number of false negatives all other metrics are low, the performance of this previous approach is considered no better than a random guess do to its low accuracy of 53.14%.

A particular analysis subject was the precision vs. distance correspondence because of the decreasing point density of distant objects makes it difficult to interpret shapes. The number of misclassified examples using 1214 static-only samples, including 300 pedestrian examples, is shown in Table VI. We can observe that there is no significant increase in the number of false positives and false negative as the distance increases. The misclassified percentages by category is also displayed. The error percentage grows for distances greater than 10 meters, although this is mainly due to the small number of actual examples in those categories. This indicates that the training data set must include more examples of long distance subjects to get more accurate results at distances longer than 10 meters.

TABLE V
CONFUSION MATRIX COMPARISON ON NEW DATA SET

	Previous Approach Detected Label		**Our Approach** Detected Label	
True Label	Human	Non-Human	Human	Non-Human
Human	2 (1.10%)	179 (98.89%)	171 (94.47%)	10 (5.52%)
Non-Human	0 (0%)	201 (100%)	9 (4.47%)	192 (95.52%))

TABLE VI
MISCLASSIFICATION ERRORS AT VARIOUS DISTANCES

Distance (m)	Total Samples	Misclassified Samples	Error Percentage (%)
0 - 1	10	0	0.00
1 - 2	58	2	3.44
2 - 3	86	6	6.97
3 - 4	81	8	9.87
4 - 5	44	0	0.00
5 - 6	60	0	0.00
6 - 7	41	5	12.19
7 - 8	307	2	0.65
8 - 9	161	2	1.24
9 - 10	186	4	2.15
10 - 11	150	13	8.66
11 - 12	19	7	36.84
12 - 13	3	3	100.0

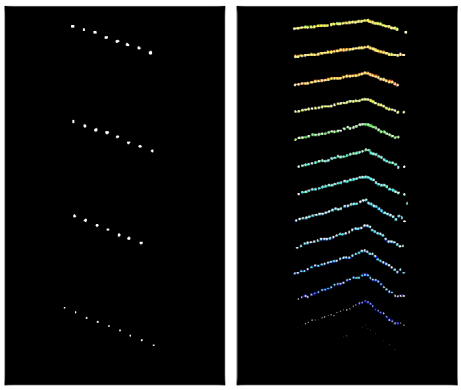


Fig. 5. Examples of clusters misclassified as humans.

After concluding that the increasing distance from sensor is not the greatest factor for false labeled samples, the misclassified data was furthered examined. A number of mislabeled examples presented similar characteristics. Two of the most common point cloud examples causing false positive errors are shown in Fig. 5. The sample shown in the right is a simple 90° corner of the inside of a room. The 90° angle of the points can fit on to a circle without much error while providing a radius measurement resembling that of a person. The sample on the left side of Fig. 5 is a small piece of wall. The wall fragments classified as humans all have exactly 4 ring lines, six to ten points in each line and are in a range from 0 to 3 meters. This samples have a limited number of points which makes shape differentiation difficult between them and a partially seen person. Both these cases involve flat surfaces, by adding a surface normal gradient feature, flat surfaces with no surface change as well as corners with drastic 90° surface changes would be discarded. The second case error could be be reduced by elevating the number of points required to be considered a valid object, as differentiating shapes with so little available information is difficult and imprecise.

VI. Conclusion

This paper describes a simple method for recognizing pedestrians from vertically sparse point cloud data. The experimental results verify that our approach deals with partially seen pedestrians, as well as decreasing point density up to 10 meters distance from the sensor. The proposed feature set is demonstrated to be effective at classifying static humans without the need for motion features. Our work can be improved by selecting additional features to deal with specific error cases.

Acknowledgment

This work was supported by the Instituto Tecnológico y de Estudios Superiores de Monterrey, the robotics education company Probaco and by The Robotics Institute through the RISS program.

References

[1] Dalal, N., and Triggs, B. "Histograms of oriented gradients for human detection". In IEEE Conf. on Comp. Vis. and Pat. Recog.(CVPR). 2005.
[2] Leibe, B.; Seemann, E.; and Schiele, B. "Pedestrian detection in crowded scenes". In IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR). 2005.
[3] T. Xu et al.A fast and robust pedestrian detection framework based on static and dynamic information. In IEEE International Conference on Multimedia and Expo. 2012.
[4] Spinello, L.; Triebel, R.; and Siegwart, R. "Multimodal people detection and tracking in crowded scenes". In AAAI Conf. on Artif. Intell. (AAAI). 2008.
[5] Arras, K., Mozos, O., Burgard, W. "Using boosted features for the detection of people in 2d range data". In IEEE International Conference on Robotics and Automation, ICRA. 2007.
[6] Spinello, L., Arras, K. O., Triebel, R., Siegwart, R. "A layered approach to people detection in 3d range data". In Proc. of the AAAI Conference on Artificial Intelligence: Physically Grounded Track. 2010.
[7] Navarro-Serment, L. E., Mertz, C., Hebert, M. "Pedestrian detection and tracking using three-dimensional ladar data". The International Journal of Robotics Research. 2010.
[8] K. Kidono, A. Watanabe, T.Naito and J. Miura. "Pedestrian Recognition Using High-definition LIDAR". Intelligent Vehicles Symp. IEEE. 2010.
[9] Hsueh-Ling Tang, Shih-Che Chien, Wen-Huang Cheng,Yung-Yao Chen, and Kai-Lung Hua, Multi-cue pedestrian detection from 3d point cloud data, in Multimedia and Expo (ICME). IEEE. 2017.
[10] Shaoshuai Shi, Xiaogang Wang, Hongsheng Li. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud". In IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR). 2019.
[11] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learningfor point cloud based 3d object detection". In CVPR, 2018.
[12] M. Yoshioka, N. Suganuma, K. Yoneda, and M. Aldibaja."Real-time object classification for autonomous vehicle using LIDAR". In International Conference on Intelligent Informatics and BiomedicalSciences (ICIIBMS), pages 210211. 2017.
[13] C. Savtchenko J. and Spletzer. "Sidewalk-leve people tracking with a low-cost 3D LIDAR system". Lehigh University Technical, Report LU-CSE-11-003. 2011.
[14] R. E. Schapire and Y. Singer, Improved boosting algorithms using confidence-rated predictions. Mach. Learn., vol. 37, no. 3, pp. 297336. 1999.
[15] Mertz, C., Navarro-Serment, L.E., MacLachlan, R., Rybski, P., Steinfeld, A., Supp, A., Urmson, C., Vandapel, N., Hebert, M., and Thorpe, C. "Moving object detection with laser scanners". J. Field Robot. 2013.
[16] Gander W., Golub G.H., and Strebel R. "LeastSquares Fitting of Circles and Ellipses BIT Numerical Mathematics". 34(4) pp. 55857. 1994.
[17] G. Louppe, Understanding Random Forests: From Theory to Practice, PhD Thesis, U. of Liege. 2014.
[18] Navarro-Serment, L. E., Mertz, C., Vandapel, N. and Hebert, M. "LADAR-based pedestrian detection and tracking". 1st. IEEE Workshop on Human Detection from Mobile Platforms, Pasadena, CA, May 20, 2008.

Using Similarity Measures to Detect Organizations in Online Escort Advertisements

Carl Edwards[1], Anthony Wertz[2] and Artur Dubrawski[3]

***Abstract*— Sex trafficking is a substantial problem in the world, and millions suffer from forced sexual exploitation. Due to the proliferation of internet, a significant amount of potential information for detecting trafficking behavior is now available online. By using this online data to trace organizations over time, law enforcement can both target specific trafficking organizations as well as better understand and address changes in trafficking activity. Past work has extracted various features from the data and attempted to cluster and classify it. In this paper, we approach this problem with several similarity measures in order to detect and monitor organizations in escort advertisement data. Our framework allows for easy incorporation of new similarities as well. Additionally, we examine multiple modalities (text advertisements and corresponding images) to enhance the detection of these trends by providing multiple perspectives. This work finds that organizations can be detected which show the evolution of advertisements temporally and geographically even as both names and phone numbers change.**

machine learning, escort advertisements, similarity measures, trend detection, multimodal data

I. Introduction

Human trafficking is a pervasive and global problem today. In 2017, the International Labour Organization estimated that 40 million people globally were victims of modern slavery. Among these, "3.8 million adults were victims of forced sexual exploitation and 1.0 million children were victims of commercial sexual exploitation in 2016" [1]. Due to the massive social impact of the internet, advertising for sex trafficking has moved online in the form of social networking sites and online classifieds [2]. In order to avoid detection by law enforcement, trafficking organizations attempt to blend in with non-trafficking related advertisements. They may periodically change phone numbers or other identifying features. Essentially, connecting advertisements becomes a game of whack-a-mole. An organization might pop up with a phone number in one location and then somewhere else with another number. By using multiple similarity measures, we are able to draw connections between advertisements even if these identifying features changes.

In 2014, DARPA launched its Memex program with a specific focus on combating human trafficking; this three year program resulted in a significant amount of research involving information retrieval, feature extraction, and classification and clustering to obtain and utilize data from this source of online information [3]. To best use this information, approaches have included training classifiers [4], [5], entity resolution [6], and graph-based techniques [7], [8] Additionally, some techniques go beyond text and incorporate multiple modalities of data using smaller supervised datasets such as in [4].

Recent work has focused on unsupervised natural language processing techniques such as word and document embeddings in order to extract and match templates for organizations [9] or to indicate sentences likely to be related to human trafficking [10].

Prior work has primarily focused on text or images, however, our technique allows for easy integration of new similarity measures from different modalities. Additionally, past efforts have frequently avoided the use of pairwise similarities since it is computationally intractable to compute them between all data points.

In this paper, we investigate the usage of multiple pairwise similarity measures to find trends and connections between advertisements. We examine text similarity based on word embeddings, similarities based on features extracted from the data such as phone numbers, names, and image hashes, and similarities based on face recognition. Additionally, we incorporate geospatial and temporal information into our framework. Like [9], we define an organization as a singular individual or group of related indivudals posting about escort services on backpage.com. Our approach would allow law enforcement officials to look for and monitor organizations suspected of sex trafficking and then build a case against them as they evolve and change over time. By combining multiple similarity measures, we are able to better characterize these organizations.

II. Methodology

A. Dataset

The dataset, D, consists of roughly 40 million advertisements which were scraped from the escorts section of backpage.com from September 2012 to December 2017. Each advertisement contains text, location, time, and images. There are approximately 20 million unique images present in the data and 562 unique locations, such as New York City. The text portion of the data is very noisy; emojis and mispellings are frequently used, words run together, and grammatical rules are ignored. Additionally, the dataset does not have labels due to its size and the domain expertise required to estimate an advertisement's likelihood of being posted by traffickers.

[1]Carl Edwards is a senior student at the Department of EECS at the University of Tennessee cedwar45@utk.edu

[2,3]Anthony Wertz, Artur Dubrawski are in the Auton Lab, Robotics Institute, Carnegie Mellon University awertz@cmu.edu; awd@cs.cmu.edu

B. Similarity Measures

Similarities measures take two points, $q, d \in D$, and compute the similarity $f : D \times D \rightarrow \mathbb{R}$. In this work, we use similarity techniques which have a range of $[0, 1]$ where q and d are more related if $f(q, d)$ is nearer to 1 and less related if $f(q, d)$ is nearer to 0.

- Text similarity: We determine text similarity between advertisements using unsupervised word embeddings. In recent years, word embeddings which use distributed representations have significantly improved state-of-the-art results on a variety of NLP tasks [11], [12]. In particular, we use fastText embeddings [13], which extend `word2vec` [11] to allow for the incorporation of subword information; this is desirable for representing this noisy text data where characters such as emojis may replace a single letter in a word and mispellings are frequent. Additionally, this model is capable of producing embeddings for unseen words, which could allow new ads to be easily embedded by an already-trained model. A fastText model is initially trained using a skip-gram architecture [11] and default hyperparameters on a corpus consisting of the text body of every ad in the dataset. It creates 100-dimensional embeddings for the vocabulary in the corpus. Following this, paragraph embeddings for each ad are created by taking the average of the word embeddings for each word in the ad. According to [14], "simply averaging word embeddings of all words in a text has proven to be a strong baseline or feature across a multitude of tasks." Cosine similarity is used to compute a value between -1 (least similar) and 1 (most similar) to measure the similarity between two embeddings. This is the only similarity which outputs values below 0. However, empirical results produce only positive values. Cosine similarity [15] between two vectors v_1 and v_2 is defined as follows:

$$similarity(v_1, v_2) = \frac{< v_1, v_2 >}{\|v_1\|\|v_2\|}$$

- Common-Feature similarity: This boolean similarity measure is 1 if two advertisements have a feature in common (e.g. a phone number) and 0 otherwise. We use it for the following similarities:
  - Phone Number Similarity: Phone numbers are extracted using regular expressions.
  - Image Hash Similarity: Images are hashed for each advertisement. This allows detection of image reuse between advertisements.
  - Name Similarity: Names are extracted using the **AnonymousExtractor** regex from [6].
- Face similarity: Faces are extracted and processed into 128-dimensional vectors using a pipeline of dlib pretrained models [16]. Face detection is performed using the CNN-based model `cnn_face_detection_model_v1`. This is chosen over the HOG (histogram of oriented gradients) model since it can be accelerated on a GPU. Next, a 5-point landmarking model trained on the dlib 5-point face landmark dataset is used to localize the faces. Finally, `dlib_face_recognition_resnet_model_v1` is used to create a 128-dimensional embedding for each detected face. This model is a ResNet based on [17] with only 29 convolutional layers instead of 34 and with the number of filters reduced by half. Two faces are considered to belong to the same person if the Euclidean distance between them is less than 0.6; this produces a binary output [16]. When using the HOG model instead of the CNN for face detection, this pipeline achieves a reported accuracy of 99.13% on the Labeled Faces in the Wild dataset [18]. The CNN-based model which we use, however, is reported to be more accurate than HOG [16]. Two advertisements are considered similar if they both have images containing the same face.

C. Pipeline

- Initially, the data is processed and textual features such as phone numbers are extracted using regex.
- A fastText model is trained on message body text, and paragraph embeddings are calculated by averaging constituent word embeddings.
- Faces are detected in the associated images and face embeddings are computed. This completes the extraction of features from the data.
- First, we optionally restrict our dataset to a specific location, such as New York City. We use an initial advertisement, or query point, to look for related advertisements. This is necessary since computing the pairwise similarity between all the data points is computationally intractable.
- Following this, we use a similarity function in order to isolate a smaller subset, $S \subseteq D$, of relevant advertisements. In this work, text similarity is used to create this subset. The text similarity is calculated between our query point and all other paragraph embeddings. A subset of advertisements which are above a certain threshold are selected as S.
- Next, other similarity measures, f, such as common-feature similarities and face similarity are used to link advertisements within this subset. For all $d_1, d_2 \in S$, d_1 and d_2 are linked if $f(d_1, d_2)$ is above some threshold for that similarity measure. This produces a temporal trail which links advertisements as they change over time.

D. Thickness of Tail

As a potential indicator of whether a query has produced an interesting trend, we use the thickness of the tail of the distribution of text similarities as a proxy representing signal vs noise in the subset, S, of text related to the query point, q. We calculate this as the ratio of points above two different similarity thresholds, the numerator threshold, t_n, and the denominator threshold, t_d. The number of ads above these two thresholds are counted: n_n and n_d respectively.

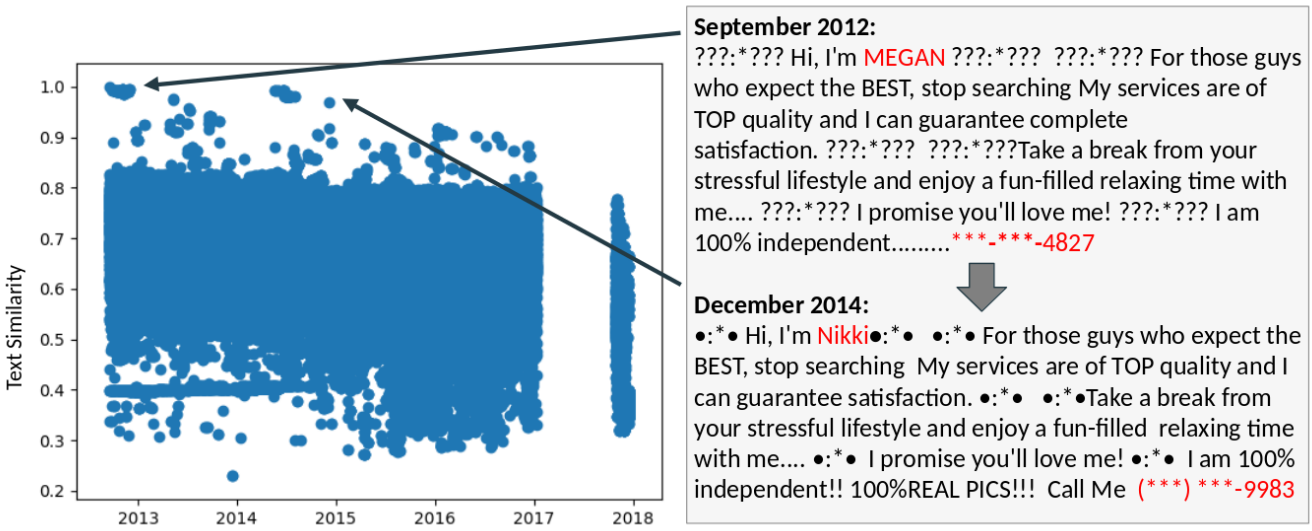


Fig. 1: Each point represents an advertisement. The y-axis is the text similarity between the query point and each ad. The query point can be seen at 100% similarity in the upper left of the figure. This organization is found within 247,000 advertisements in Pittsburgh. Activity ceases after a couple months and resumes two years later with a different name and phone number.

We then calculate our thickness, $\Theta = \frac{n_n}{n_d}$. A high Θ value indicates that a large portion of the closely related ads are very closely related in text similarity: the tail gets thicker after t_n than it does at t_d.

III. Results

A. Text Similarity

We find that cosine similarity between unsupervised paragraph embeddings can be used to successfully extract advertisements related to the query point shown over time. Most advertisements fall in a range of 0.4 to 0.8. Advertisements over a certain threshold are considered related to the query point. We empirically determine this threshold as 0.95. We find that advertisement text is frequently reused, even over several years and when features like phone numbers and names change. The evolution of advertisements can also be observed, which can be seen in Figure 2 where the same organization modifies an advertisement from 2012 to 2016; the text similarity slowly decreases as the ad is continuously tweaked. Figure 1 shows an example of related ads in Pittsburgh which would not be linked by any of these features alone. Prior work often uses phone numbers as an oracle label for groups. Unfortunately, this approach can fail to successfully capture an entire organization. For example, Figure 3 showcases an organization which periodically changes its phone number. Additionally, many phone numbers are obfuscated, such as in Figure 6, which poses another challenge to using phone numbers to predict organizations.

Although text similarity can successfully be used to find related advertisements, false positives due to high text similarity to the query point are erroneously included in the subset of textually related ads. This creates noise which can make isolating organizations more difficult. Figure 5 shows an organization which is surrounded by this noise. Noise is much more common if a specific location is not selected to reduce the size of the subset. This is due to an increase in the magnitude of advertisements from a few hundred thousand in an individual city to the entire dataset of 40 million ads.

B. Linking

In order to address the noise in the subset of advertisements selected using text similarity, we isolate organizations from the noise by using additional similarity measures to link advertisements together. Since the non-text similarity measures we use are boolean in nature, we link ads if the similarity between them is 1. Additionally, this serves to help connect advertisements into organizations which may change names, phone numbers, and other features at various intervals of time. For example, Figure 3 show an organization which can be linked using text, name, and phone similarity. Figure 4 shows an example where an organization is isolated from noise using other similarity measures.

C. Face Similarity

In order to augment the text similarity and similarities based on features extracted from the text, we incorporate multiple modalities of the data by incorporating visual information in the form of faces. Previous studies, such as [6], only use image hashcodes. While hashcodes are very useful, they don't incorporate visual information which may help tie together organizations. Figure 4 shows how faces connect two groups of ads which image hashcodes do not. The top-left organization completely changes all its images in 2013, but they still share the same face.

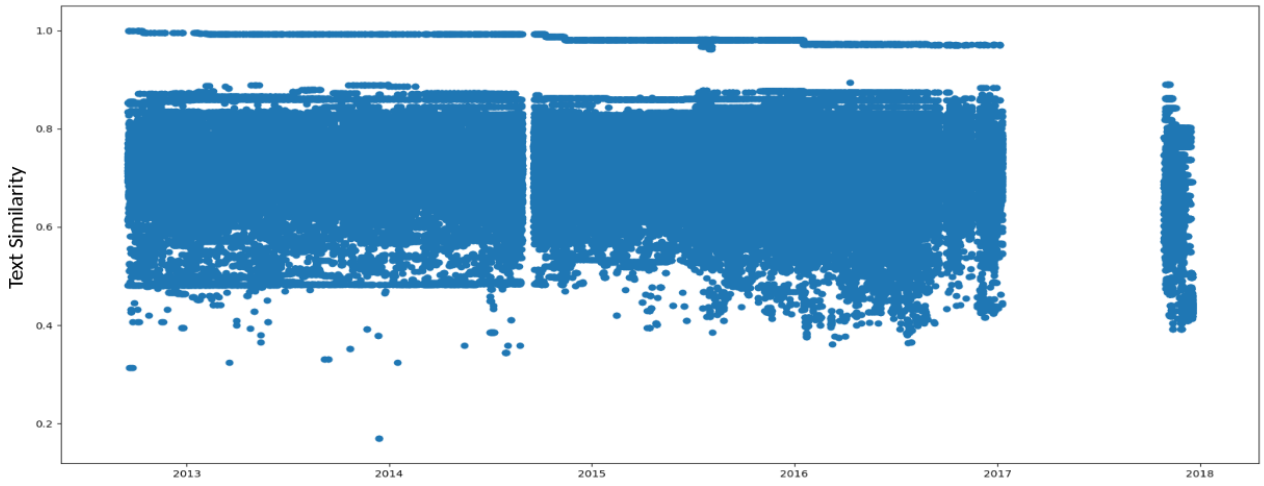


Fig. 2: This figure also shows another organization within Pittsburgh. An ad is posted over 2,000 times and continuously tweaked. Notably, the names of the organization's apparent pimps change. This organization was actually originally identified in [19] during 2012.

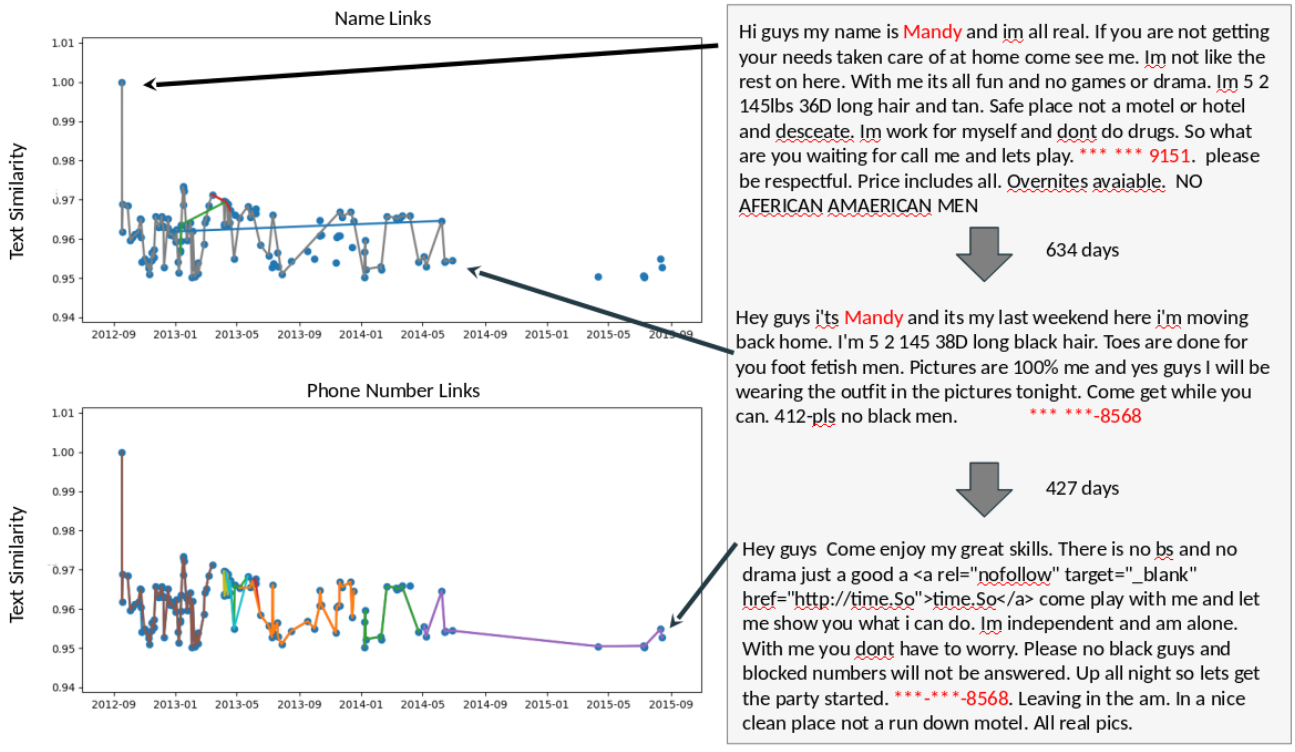


Fig. 3: An organization from Pittsburgh is linked together over several years by combining phone number and name similarities. Each colored line represents a shared attribute. For example, in the top plot the name 'Mandy' is connected using a gray line. The bottom plot shows periodic changes in phone number.

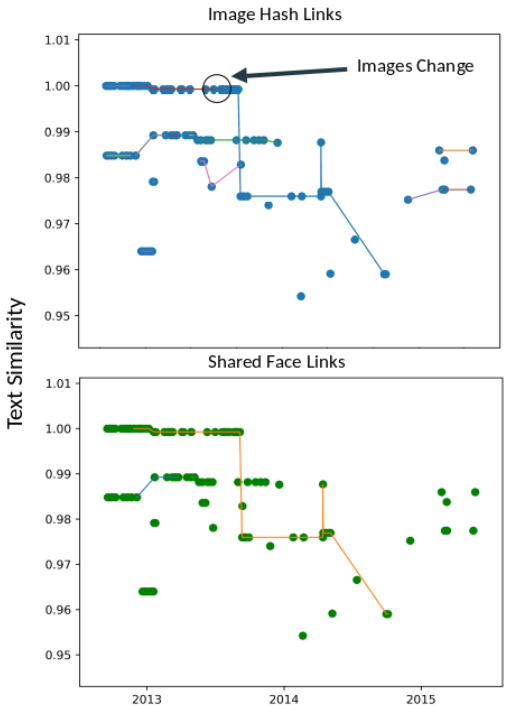


Fig. 4: This figure shows an organization (top left in plot) in LA which changes all its images at once. Face similarity is used to connect these two groups of ads with separate image hashes.

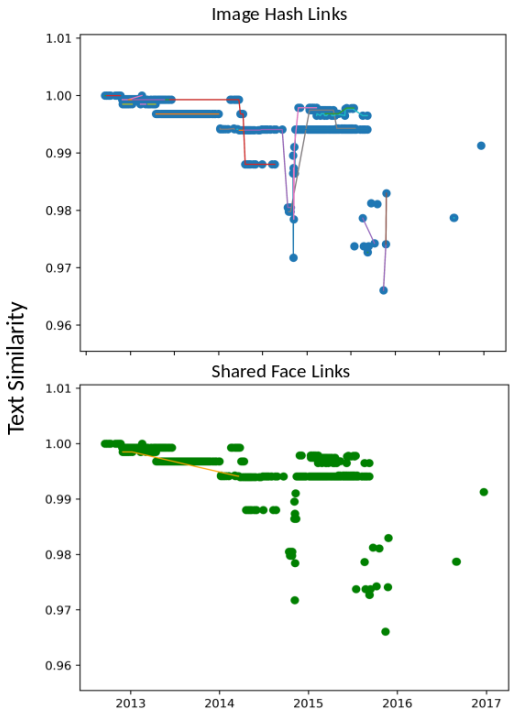


Fig. 5: Although these two organizations in LA are connected by faces, manual inspection shows that this is a false positive.

Fig. 6: This figure shows a query of all the data which finds a multi-city operation in several locations around the New York City metropolitan area. The text from one of these advertisements is shown above. The ads have location-specific URLs and obfuscated phone numbers.

In this work, we extract approximately 635,000 faces from the first 5 million chronologically-posted unique images in the dataset. Due to time constraints and computational limitations, we were unable to process the remaining images. However, images which may have been initially posted in 2012 can often be found during later years, since images are frequently reused. The usage of stock images is also common.

Although using face recognition allows improved connections between ad images, the face recognition model appears to suffer from false positives. This may be due to irregular poses, occlusions, and intentional obfuscation. Additionally, the face recognition model may not perform as well on minorities. Figure 5 shows an example of two organizations which are linked using both text and face similarity, However, inspection shows they are not related. Due to privacy concerns, an example of matching faces which are false positives are not shown.

D. Multiple Locations

In addition to searching for organizations within specific cities, nationwide searches can also be conducted to find multi-city organizations, such as in Figure 6. In fact, the URLs in these ads appear to be subdomains of an organization taken down by the DoJ in early 2019 [20]. Unfortunately, searching all locations results in more noise being included in the textually related subset of advertisements due to the increase in magnitude of data. Since nationwide similarity computations are more expensive, we find it prudent to examine trends occurring within a specific location first before looking for multi-city activity. This is supported by [6]'s finding that location is the most informative feature for their entity resolution classifier.

E. Examining Thickness of Tail

We also find that the thickness of the tail of a distribution of text similarities can be indicative of an organization. An example of this can be seen in Table I. After examining this

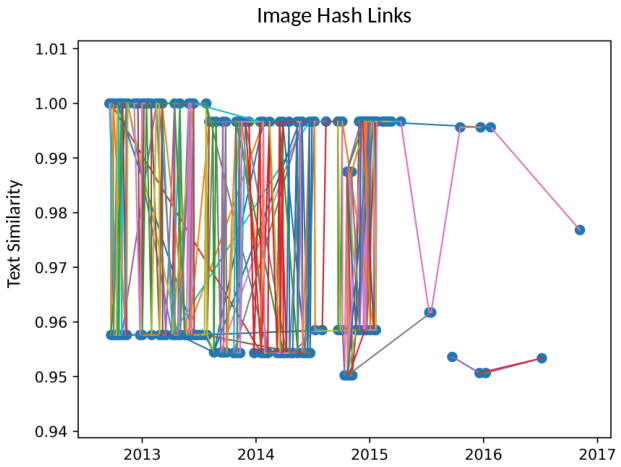


Fig. 7: An organization identified in data from Los Angeles. Its query point corresponds to index 13 in the Table I, which has a Θ value of 47.7

table, we then selected index 13 for investigation, resulting in Figure 7. It is important to note that although this metric indicates the likely existence of an organization, a thin tail does not indicate that no organization exists. In queries with more text noise, several unrelated organizations could appear. Additionally, there might be related ads whose text similarities are not above the threshold t_n.

Index	n_d	n_n	Θ
4	2451	64	2.61
5	1567	23	1.47
6	1980	19	0.96
8	330	153	46.36
9	410	1	0.24
11	988	929	94.03
12	4818	44	0.91
13	153	73	47.71
15	747	352	47.12
16	572	1	0.17

TABLE I: An example of comparing queries using the thickness of tail on a few advertisements in Los Angeles. Each row represents an ad with over 50 textually related ads taken from the first 20 ads. n_n is the number of ads above a similarity threshold of 0.99 and n_d is the number above a threshold of 0.95. Θ is the thickness of tail, $\frac{n_n}{n_d}$.

IV. Conclusion

In this paper, we examine the use of multiple similarity measures in order to find trends and connections indicating organizations within escort advertisements. We demonstrate how this technique can find organizations even as they change names and phone numbers. Additionally, the use of multiple similarities can help to remove erroneous noise from being included in potential organizations. In particular, this framework allows easy incorporation of new similarity techniques, and we leverage this capability in order to draw information from multiple modalities of the data, which can help provide a more complete picture of an organization. This would allow law enforcement to investigate for the existence of sex trafficking organizations, monitor organizations as their activity changes (such as attempting to conceal their identity), and build case evidence against them.

V. Future Work

In the future, our technique can be improved through the use of more features from [5] as well as incorporation of other similarities. The usage of visual information can be improved through similarities based on background and foreground segmentation and matching. Additionally, the use of a weighted average-based paragraph embedding and higher dimensional embeddings may improve the quality of the measure of text similarity.

Acknowledgment

The authors would like to thank the members of the Auton Lab and Robotics Institute Summer Scholars program 2019 for all their insightful help and advice. Additionally, we would like to acknowledge the support of the NSF REU program.

References

[1] I. L. Organization, "Global estimates of modern slavery: Forced labour and forced marriage," 2017.

[2] M. Latonero, "Human trafficking online: The role of social networking sites and online classifieds," *Available at SSRN 2045851*, 2011.

[3] K. Hundman, T. Gowda, M. Kejriwal, and B. Boecking, "Always lurking: Understanding and mitigating bias in online human trafficking detection," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 2018, pp. 137–143.

[4] E. Tong, A. Zadeh, C. Jones, and L.-P. Morency, "Combating human trafficking with deep multimodal models," *arXiv preprint arXiv:1705.02735*, 2017.

[5] A. Dubrawski, K. Miller, M. Barnes, B. Boecking, and E. Kennedy, "Leveraging publicly available data to discern patterns of human-trafficking activity," *Journal of Human Trafficking*, vol. 1, no. 1, pp. 65–85, 2015.

[6] C. Nagpal, K. Miller, B. Boecking, and A. Dubrawski, "An entity resolution approach to isolate instances of human trafficking online," *arXiv preprint arXiv:1509.06659*, 2015.

[7] R. Rabbany, D. Bayani, and A. Dubrawski, "Active search of connections for case building and combating human trafficking," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2120–2129.

[8] P. Szekely, C. A. Knoblock, J. Slepicka, A. Philpot, A. Singh, C. Yin, D. Kapoor, P. Natarajan, D. Marcu, K. Knight *et al.*, "Building and using a knowledge graph to combat human trafficking," in *International Semantic Web Conference*. Springer, 2015, pp. 205–221.

[9] L. Li, O. Simek, A. Lai, M. Daggett, C. K. Dagli, and C. Jones, "Detection and characterization of human trafficking networks using unsupervised scalable text template matching," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 3111–3120.

[10] M. Kejriwal, J. Ding, R. Shao, A. Kumar, and P. Szekely, "Flagit: A system for minimally supervised human trafficking indicator mining," *arXiv preprint arXiv:1712.03086*, 2017.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[14] T. Kenter, A. Borisov, and M. De Rijke, "Siamese cbow: Optimizing word embeddings for sentence representations," *arXiv preprint arXiv:1606.04640*, 2016.
[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[16] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[18] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008.
[19] E. Kennedy, "Predictive patterns of sex trafficking online," *Dietrich College Honors Theses*, 2012.
[20] "Nationwide sting operation targets illegal asian brothels, six indicted for racketeering," Jan 2019. [Online]. Available: https://www.justice.gov/usao-or/pr/nationwide-sting-operation-targets-illegal-asian-brothels-six-indicted-racketeering

Angle-Specific Goal Assignment for Multi-UAV Persistent Coverage

Daniel Feshbach[1], Shohin Mukherjee[2] and Maxim Likhachev[3]

***Abstract*— In a variety of situations from traffic monitoring to disaster response, mobile robots present promising tools to observe large areas over time. Planning for *persistent coverage* with one or more robots involves ensuring that each site of interest is observed at least as frequently as some given minimum, which may vary by site. This paper builds on [1], which provides an online kinodynamic planning framework for persistent coverage of a grid-based environment using multiple robots with circular sensor footprints. Since its sensor footprint is a circle, the coverage from position (x, y) does not depend on the angle the robot is facing, so the framework assigns goal positions as (x, y) and then finds plans to reach that position indifferent to the final angle. However, many sensors in the real world face a specific direction and thus have coverage which depends on the robot's heading, requiring planning systems to specify heading in goal assignment. This paper extends the planning framework from [1] to non-circular sensor footprints by assigning and planning to angle-specific (x, y, θ) goal poses. Most of our work is in the goal assignment, which balances the the urgency of what is covered at the goal pose with the feasibility (approximated using Dubins Distance) of reaching that specific pose.**

I. Introduction and Related Works

In robotics, the problem of *persistent coverage* involves using one or more mobile robots to continually monitor a region over time [1]. It is characterized by the need to repeatedly re-observe the same areas as frequently as possible or based on some desired minimum frequency. Desired observation frequencies may vary between regions in the environment, with some at higher priority than others.

This paper builds on [1], which presents a planning system for persistent coverage of a grid-based environment with multiple unmanned aerial vehicles (UAVs) which cover a specified sensor radius around their position. It approaches the problem by running a continuous loop which plans for one UAV at a time for upcoming window of time. The *goal assigner* selects a target position which balances close proximity to the current position with high urgency of the goal position's coverage, based on how long remains until the cells will have been unobserved for too long. Then the *goal planner* finds a collision-avoiding, kinodynamically feasible plan of motion primitives to reach this goal.

Since [1] models the sensor footprint as a circle centered on the robot, what a UAV covers does not depend on the angle it is facing. Its goal assigner therefore outputs (x, y) *positions* on the flight plane, and its goal planner seeks paths which end at this position facing any angle. However, the assumption of a circular sensor footprint does not hold for many real-world sensor systems, particularly on non-holonomic or kinodynamically constrained vehicles like UAVs. In such systems, the coverage depends on which angle the robot is facing, so that angle must be considered in planning systems.

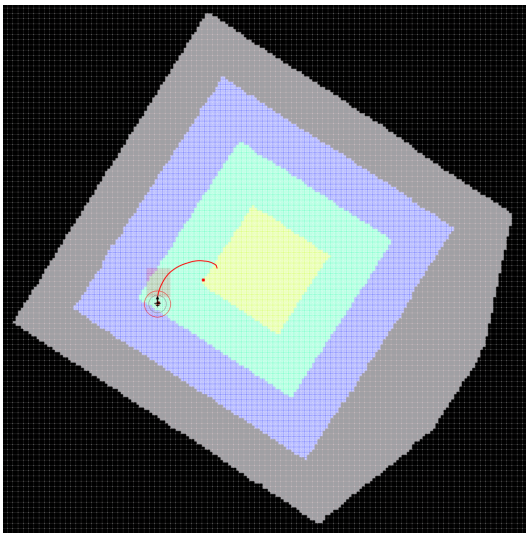


Fig. 1: A UAV with a plan to a goal. The black center of the circle is the UAV's current location, and it is currently covering the red rectangle in front of it. The red point is the location of the UAV's current navigation goal, and the red curve is the path that takes it within a distance tolerance of the goal. The goal angle faces towards the yellow inner square, because that is the highest priority region to cover.

This paper extends the planning framework from [1] to non-circular sensor footprints by assigning and planning to angle-specific (x, y, θ) goal *poses* instead of just (x, y) *positions*. Specifically, we consider a rectangular footprint displaced in front of the robot. This is more difficult than using a circular sensor footprint because the space of possible goals is much larger, and the planner must reach a state which satisfies a much more specific restriction. We approximate the feasibility of reaching goals with specific angles using the length of the shortest Dubins path [2].

To restrict the size of the search space, we select a single heading from each cell: we test and compare two versions of how to select the heading from each cell. One

[1]Daniel Feshbach is a senior undergraduate in the Department of Computer Science, Haverford College, Haverford, PA 19041, USA danielfeshbach@gmail.com

[2]Shohin Mukherjee is a PhD. student in the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA shohinm@andrew.cmu.edu

[3]Maxim Likhachev is with the Search-Based Planning Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA mlikhach@andrew.cmu.edu

version optimizes for urgency of a pose's coverage, the other minimizes the Dubins path from the current pose. Then, in the space of each cell associated with its 'best' angle, we conduct a multi-Goal Dijkstra search to minimize a cost function which accounts for both urgency and feasibility.

Simulation experiments demonstrate that assigning and planning to angle-specific goals improves coverage performance compared to approximating rectangular sensor footprints as circles. We also find that, when selecting the angle from each cell to consider in the goal assignment search, minimizing the Dubins distance successfully leads to assigning goals which are easy to reach. This improves the success rate of the motion planning search and resulting in superior coverage performance.

Section II formalizes the problem statement and provides a detailed summary of system structure (based on [1]) in which we work. Section III explains our version of the goal assigner, the adjustments to which are this paper's central contribution. Section IV discusses how we adjusted the goal planner, section V presents our experimental setup and results, and section VI discusses concluding thoughts and future directions.

II. System Setup and Problem Statement

We consider n UAVs $\{U_1, \ldots, U_n\}$ flying over a 2D environment and observing rectangular regions in front of them. To reduce the dimensions of path planning, we assume all UAVs fly at a constant height. As in [1], the task of the overall system is to observe environment cells at least as often as a frequency specified in each cell, so we use the same overall system structure. This paper considers the problem of how to adjust the [1] system to handle sensor footprints which are not circular, and whose coverage subsequently depends on the angle the UAV is facing.

A. Environment

We discretize the environment into a *mission-map* grid M. Let $c_{i,j}$ denote the cell at row i and column j of M. There are three types of cells:

1) Coverage cells, which UAVs can fly over and are tasked with observing repeatedly. Let M_C denote the set of all coverage cells in M.
2) No-coverage cells, which UAVs can fly over but need not observe. Let M_{NC} denote the set of all no-coverage cells in M.
3) Obstacle cells, which UAVs cannot fly over, must avoid colliding with, and need not observe.

Let $M_{\text{fly}} = M_C \cup M_{NC}$, the set of all cells which the UAV can fly over.

Each coverage cell $c_{i,j}$ has a specified *lifetime* $\ell(c_{i,j})$, a desired upper bound on the time since the cell has last been covered. While the system is running, we keep track of the *age* $a(c_{i,j})$ of each coverage cell, the time since it has last been covered. We say a cell *expires* when its its age exceeds its lifetime, and that its *time remaining* is $\ell(c_{i,j}) - a(c_{i,j})$.

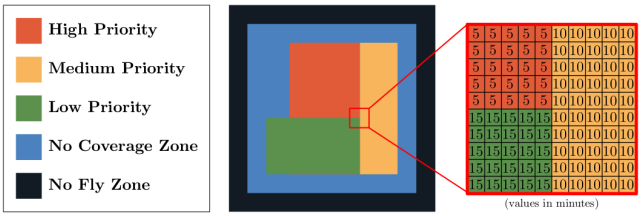


Fig. 2: From [1], an example environment. The red, yellow, and green zones are coverage cells with lifetimes of 5, 10, and 15 minutes respectively. The blue zone is no-coverage cells, and the black border is obstacle cells.

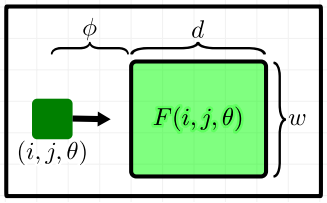


Fig. 3: The dimensions of the UAV sensor footprint. For our experiments we use $d = 30$ meters, $w = 25$ meters, $\phi = 10$ meters, on grid cells of size 3 meters.

B. UAVs

Each UAV U_k is a kinodynamically constrained system, and so we use the motion planner from [1] which plans using kinodynamically feasible motion primitives defined on states which include velocity. The contribution of this paper focuses on the UAV's pose (i, j, θ), where (i, j) is its coordinates in the grid and θ is the angle it faces.

Each UAV is equipped with a sensor whose footprint F_k is a rectangle of width w and length d in front of the robot. The center of the rectangle is at distance ϕ in direction θ from (x, y). A cell in the environment is *covered* by U_k when any portion of it is within F_k. Whenever a coverage cell is covered, its age is reset to 0 and its time remaining is reset to its lifetime.

C. Persistent Coverage Problem

The goal of the system is to maintain the highest possible level of coverage of the environment relative to the cell lifetimes. Specifically, we seek to minimize the number of cells which are *expired* (have their age exceed their lifetime) and minimize the environment's overall *coverage criticality*, defined as the average $a(c_{i,j})/\ell(c_{i,j})$ of all coverage cells $c_{i,j}$ at a particular time.

D. Prioritized Planning System

To approach the overall problem of persistent coverage, we use the prioritized planner (PP) from [1]. This considers UAVs one at a time, and separates out the questions of where it will go next (what we call *goal assignment*) and how it will get there (*goal planning*). Goal assignment considers the current position of the UAV and coverage status of the environment, adjusting for what other UAVs are about to cover

in their *committed plans*. Goal planning runs a weighted A* search for the goal position on a graph of kinodynamically feasible motion primitives, avoiding obstacle cells and other UAVs.

After the goal planner finds a path, the prioritized planner takes the first $t_{\max}$ seconds of the path and declares it *committed* so that the subsequent UAVs can treat it as a moving obstacle and avoid it in their goal planning. The choice of an appropriate $t_{\max}$ must be made based on the specific application, and [1] discusses the competing concerns with it being too long or too short.

[1] uses a circular sensor footprint, so it assigns goals which were simply particular cells (i, j), and plans paths which reach the cells without regard to the angle it ends up facing. Since we use a footprint which varies based on the robot's angle, this paper considers how best to assign goal poses specified as (i, j, θ). This is more challenging because the space of possible goals is larger, and because goals it substantially restricts the UAV states which qualify as reaching the goal.

III. Goal Assignment Algorithm

We select goal poses which seek to cover the most urgent cells while being as easy as possible to reach from the robot's current position. Like [1], we achieve this by conducting a multi-goal Dijkstra search on the 8-connected grid of non-obstacle cells, but our approach adjusts for the coverage and reachability of the specific angles we assign. Multi-goal Dijkstra proceeds by augmenting an existing graph with a *pseudo-goal* node, which is connected to existing nodes (the potential goal nodes) by *pseudo-edges*. It then conducts a Dijkstra search to find the shortest path from a start node to the pseudo-goal. The node immediately prior to the pseudo-goal in the shortest path - the one whose pseudo-edge is used to reach the pseudo-goal - is selected and returned as the output goal. In our case, our starting graph is the 8-connected grid of non-obstacle cells, with weights 1 and $\sqrt{2}$ for horizontal or vertical edges and diagonal edges respectively. As detailed below, we consider the angle with the most urgent coverage from each cell, and weigh the psuedo-edges by a combination of how urgent the coverage is and how difficult it is to reach that angle, so the total path length balances coverage and reachability.

A. Selecting the Optimal Angle at Each Cell

Let $F(i, j, \theta)$ be the set of coverage cells covered by the footprint at (i, j, θ). We define the *remainder* of a pose (i, j, θ) as

$$R(i, j, \theta) = \underset{c \in F(i,j,\theta)}{\text{mean}} \left(\ell(c) - a(c)\right),$$

the average time remaining over $F(i, j, \theta)$. Note that smaller remainders are more urgent to cover because the cells are closer to expiration.

Let (i_s, j_s, θ_s) be the pose of the UAV at the start of the time period for which we are selecting a goal. We use Dubins distance to approximate the feasibility of reaching a specific goal from this position, with a minimum turning radius

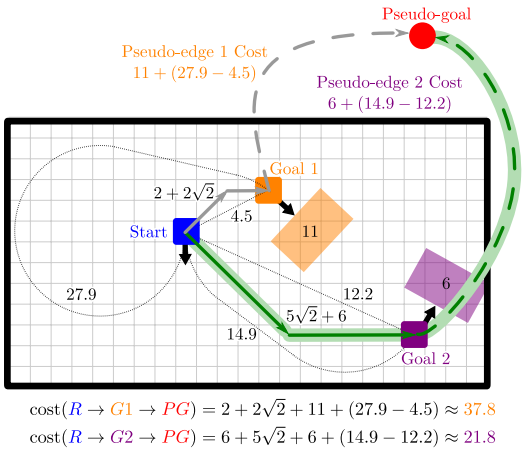


Fig. 4: Modified from [1], an illustration of the multi-goal Dijkstra search used in goal assignment. Between these two possible goals, the search will select Goal 2 because the total path cost to the pseudo-goal through it is shorter than the path through Goal 1. While Goal 1 is closer in grid distance than Grid 2, it is much farther in Dubins distance, and covers a region with a higher average time remaining until expiration.

r. This radius is selected manually based on the motion primitives used in goal planning, aiming to be reasonably representative of desirable and realistic flight paths. Given a potential goal cell $c_{i,j}$ and an angle θ, let $D(i, j, \theta)$ denote the shortest Dubins path from (i_s, j_s, θ_s) to (i, j, θ) given a minimum turning radius of r.

We discretize the unit circle into a set Θ of 16 angles, which we consider as the possible angles for goals. At each non-obstacle cell node $c_{i,j}$, we select an angle $\theta^*_{i,j}$ to associate with the cell node in the Dijkstra search. This $\theta^*_{i,j}$ is the angle we would assign if the multi-goal Dijkstra selected node $c_{i,j}$. We test two separate versions of how to select this angle:

1) Select $\theta^*_{i,j}$ as the angle which minimizes the remainder of the pose's coverage,

$$\theta^*_{i,j} = \min_{\theta \in \Theta} \left(R(i, j, \theta)\right),$$

i.e., the angle which provides the most urgent coverage from this cell.

2) Select $\theta^*_{i,j}$ as the angle with the shortest Dubins path from the start pose,

$$\theta^*_{i,j} = \min_{\theta \in \Theta} \left(D(i, j, \theta)\right),$$

i.e., the angle which seems easiest to reach.

While we prioritize one over the other when selecting $\theta^*_{i,j}$, we will account for both coverage quality and feasibility to reach in our selection of which cell node $(i, j, \theta^*_{i,j})$ to assign, by incorporating both $R(i, j, \theta^*_{i,j})$ and $D(i, j, \theta^*_{i,j})$ into the cost of the pseudo-edge from node $c_{i,j}$.

B. Balancing Distance and Time-Remaining Magnitudes

The remainder is in units of time while the rest of the path cost is in units of distance, so simply adding them together

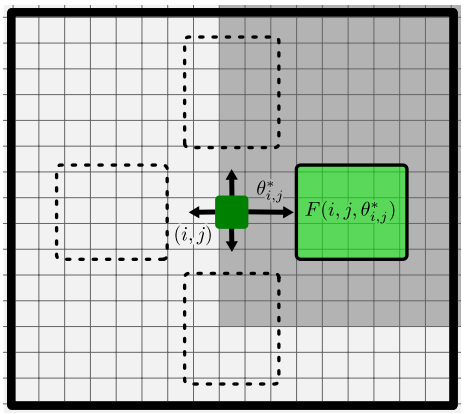


Fig. 5: Version (1), optimizing for urgency of coverage, of selecting the angle $\theta^*_{i,j}$ for $c_{i,j}$. The dark grey cells are closer to expiration than the light grey cells, so the most urgent coverage is given by having the UAV face to the right (the highlighted green footprint). This figure depicts 4 angle options for clarity, but in our experiments we consider 16.

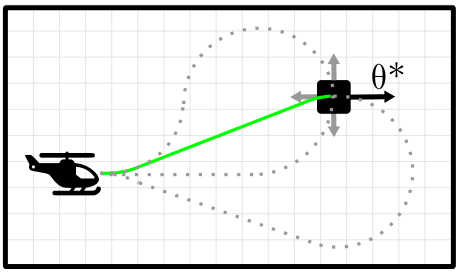


Fig. 6: Version (2), optimizing for feasibility to reach, of selecting the angle $\theta^*_{i,j}$ for $c_{i,j}$. The dotted grey paths are all longer than the solid green path, so in this version $\theta^*_{i,j}$ would be the angle to the right. This figure depicts 4 angle options for clarity, but in our experiments we consider 16.

may not produce meaningful or useful values for total path lengths. Time remaining until cell expiration and time for the UAV to travel a path are fundamentally different uses of units of time, so we cannot simply use UAV speed to convert between distance and time. Instead, we must ensure that distances and coverage urgency are each substantial contributing factors to goal selection, by scaling them to have similar magnitudes.

We achieve this by multiplying remainders by a weight W_t selected to balance the average time remaining $\overline{r_t}$ (at the current time t) over all coverage cells with the average euclidean distance $\overline{d_t}$ from the current start position to each coverage cell.

$$\overline{r_t} = \underset{c \in M_C}{\text{mean}} \left(\ell(c) - a(c)\right)$$

$$\overline{d_t} = \underset{c_{i,j} \in M_C}{\text{mean}} \left(\sqrt{(i-i_s)^2 + (j-j_s)^2}\right)$$

$$W_t = \overline{d_t}/\overline{r_t}$$

The result is $W_t\overline{r_t} = \overline{d_t}$, meaning that the average time remaining is scaled to equal the average euclidean distance. This balancing is an improvement over [1], which manually selected a weight which remained fixed throughout the run.

C. Balancing Dubins and Euclidean Distances

In order to approximate the feasibility and cost of reaching specific poses, we add a measure based on Dubins distance to our pseudo-edge costs. However, the search of the 8-connected grid already incorporates an approximation of the Euclidean distance to $c_{i,j}$ into the path length. To avoid double-counting Euclidean distance, we subtract this from our Dubins distance before adding the result to our pseudo-edge costs.

$$D'(i,j,\theta^*_{i,j}) = D(i,j,\theta^*_{i,j}) - \sqrt{(i-i_s)^2 + (j-j_s)^2}$$

D. Selecting the Optimal Pose

The total cost of the pseudo-edge from the node corresponding to cell $c_{i,j}$ is

$$P_{i,j} = W_t R(i,j,\theta^*_{i,j}) + D'(i,j,\theta^*_{i,j})$$

Let $G_{i,j}$ denote the path length in the 8-connected grid from (i_s, j_s) to (i,j). The Dijkstra's search as a whole will select the cell node $c_{i,j}$ which minimizes the total path length,

$$\min_{c_{i,j} \in M_{\text{fly}}} (G_{i,j} + W_t R(i,j,\theta^*_{i,j}) + D'(i,j,\theta^*_{i,j})),$$

and assign the goal pose $(i,j,\theta^*_{i,j})$. The chosen pose balances being quickly reachable from the start pose with covering relatively urgent regions.

Note in line 11 of Algorithm 1 that as we make a breadth-first search through the 8-connected grid, we ignore any cell nodes whose grid distance from the start cell is longer than the shortest currently-known path to the pseudo-goal. This saves time because there is no way such cells or their descendants can define a shorter path to the pseudo-goal.

IV. Goal Planner Adjustments

Our goal planner is essentially the same as [1], except that we terminate our search when we reach a state within small tolerances of both the spatial position and the angle of the goal, rather than just of the spatial position. Since this is a much more restrictive condition and thus more difficult to reach in a search, we run our experiments with a higher suboptimality bound (weight in the weighted A* search) than used in [1], and give it more time to conduct the search.

Additionally, we improve robustness over [1] to failures of the goal planner. In event that the goal planner fails to find any path to the goal within its allotted time, [1]'s default behavior is to hover in place. This happens extremely rarely when planning without a specific goal angle, but much more frequently in our more difficult search. Hovering in place leaves UAVs in essentially the same position relative to the environment, often leading them to keep choosing the same difficult-to-reach goal and remaining stuck. To prevent this, our system instead has the Goal Planner always return the last path in its search if it times out, and log an error if this path does not reach the goal.

Algorithm 1 Goal Assignment: Multi-Goal Dijkstra

$W_t \leftarrow \overline{d_t}/\overline{r_t}$
$pseudoGoal \leftarrow$ new graph node
$pseudoGoal.\text{costTo} \leftarrow \infty$
$start \leftarrow$ new graph node
$start.\text{coordinates} \leftarrow (i_s, j_s)$
$start.\text{costTo} \leftarrow 0$
$frontier \leftarrow$ queue containing only $start$
while $frontier$ is not empty **do**
  $thisNode \leftarrow$ pop front of $frontier$
  $(i, j) \leftarrow thisNode.\text{coordinates}$
  if $thisNode.\text{costTo} \leq pseudoGoal.\text{costTo}$ **then**
    $\theta \leftarrow \theta^*_{i,j}$
    $pseudoEdgeCost \leftarrow W_t R(i, j, \theta) + D(i, j, \theta) - \sqrt{(i - i_s)^2 + (j - j_s)^2}$
    $totalCost \leftarrow thisNode.\text{costTo} + pseudoEdgeCost$
    if $totalCost < pseudoGoal.\text{costTo}$ **then**
      $pseudoGoal.\text{costTo} \leftarrow totalCost$
      $pseudoGoal.\text{predecessor} \leftarrow totalCost$
    end if
    for $(i, j) \in thisNode.\text{neighboringCells}$ **do**
      if (i, j) has not been seen before **then**
        $neighbor \leftarrow$ new graph node
        $neighbor.\text{predecessor} \leftarrow thisNode$
        $neighbor.\text{costTo} \leftarrow thisNode.\text{costTo} +$ $\{\sqrt{2}$ if diagonal neighbor, 1 otherwise$\}$
        push $neighbor$ to back of $frontier$
      end if
    end for
  end if
end while
$nodeSelected \leftarrow pseudoGoal.\text{predecessor}$
$(i, j) \leftarrow nodeSelected.\text{coordinates}$
$\theta \leftarrow \theta^*_{i,j}$
return (i, j, θ)

V. Experimental Validation

A. Evaluation Criteria

Like [1], we measure system performance by evaluating environment states and timing system components over 2-hour runs. Specifically, we log the percentage $E(t)$ of coverage cells which are expired at time t as well as the *coverage criticality* $C(t)$ of the environment defined in [1] as

$$C(t) = \underset{c \in M_C}{\text{mean}} \left(\frac{a(c)}{\ell(c)} \right),$$

the average ratio of age to lifetime of all coverage cells. Ideal system behavior is to keep $E(t)$ at zero and $C(t)$ low and stable. Meanwhile, we measure the average time taken by Goal Assigner and Goal Planner respectively, as well as the success rate of Goal Planner at finding a plan which reaches the goal within its allotted 6 seconds.

We compare the two versions our system (using coverage urgency vs Dubins reachability to select $\theta^*_{i,j}$) as well as two benchmarks:

1) A version of our system which assigns and plans to goal poses with specific angles (i, j, θ), but *does not* incorporate Dubins distance at all, or otherwise in any way account distinguish between a goal being close and being easy to reach. Not only does it use coverage urgency to select $\theta^*_{i,j}$, it also does not add the Dubins-minus-euclidean distance to the pseudo-edge costs.
2) A system essentially from [1], using a circular footprint of diameter 25 meters (the width the rectangular footprint sweeps out if it moves straight forward). This assigns and plans to goals only in spatial coordinates (i, j), and maintains its internal representation of cell ages based on the circular footprint. However, we separately maintain the actual environment state based on the rectangular footprint, which is what we use to measure the performance. This measures how well [1]'s circular-footprint system works when applied on UAVs whose actual footprints are not circular, demonstrating the need for this paper's modifications. For ease of comparison, however, we do make one change from [1]: in this version we automatically and dynamically set the weight W_t between distance and time, as described in III-B

B. Experiment Setup

We test the system in an environment of size 220x220 meters, discretized into 3x3 meter cells. Pictured in Figure 1, this environment has concentric square regions featuring cells with lifetime increasing inwards out. The lifetimes are 5, 10, and 20 minutes respectively for the yellow, green, and blue regions, while the grey region on the outside is a no-coverage zone and the black is obstacle cells. We consider UAVs with rectangular footprints of width $w = 25$ meters and length $d = 30$ meters, displaced $\phi = 10$ meters in front of the UAV. Our Dubins approximation of path cost uses minimum turning radius $r = 19$ meters. We allow the Goal Planner up to 6 seconds find a path to the goal, with a sub-optimality bound of 30, and commit up to $t_{\max} = 30$ seconds of plans.

C. Results

Figure 7 plots $E(t)$ and $C(t)$ over a 2-hour run for each version, while table II shows the averages of these values over the run. Our proposed approaches maintain better coverage (indicated by lower values for $C(t)$ and $E(t)$) than the benchmark which falsely treats the footprint as a circle. The benchmark which completely ignores Dubins distance performs surprisingly well, about as well as the version which considers Dubins distance only in the pseudo-edge costs after selecting $\theta^*_{i,j}$. The version which selects $\theta^*_{i,j}$ to minimize the Dubins distance, however, performs noticeably better than the others.

Table I presents timing data for the goal assigner and the goal planner, and the frequency at which the goal planner failed to find a path to the goal. Table II shows the average duration of the plans returned by the goal planner (including

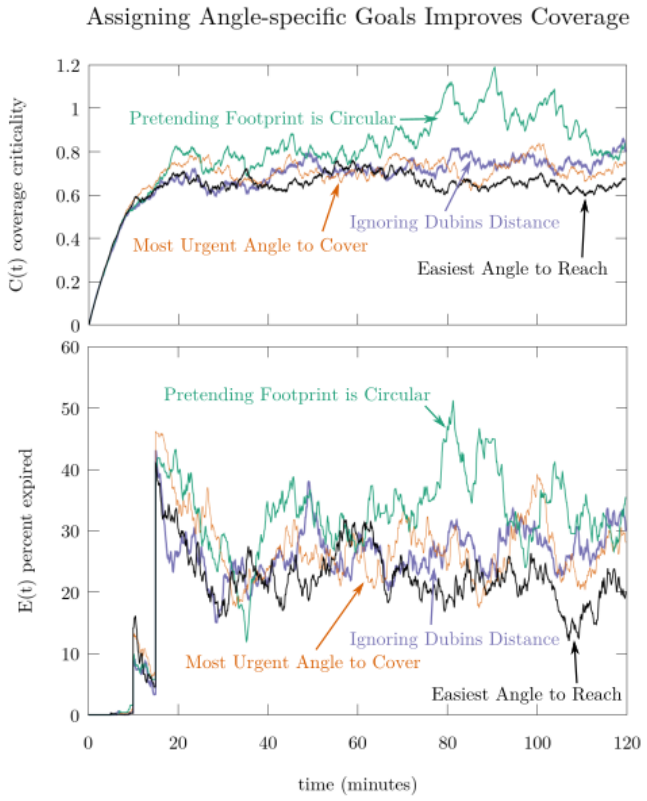


Fig. 7: Plots of $C(t)$ and $E(t)$ through a 2-hour run. For both, lower values indicate better coverage. All approaches which assign angle-specific goals outperform the version of the system which approximates the rectangular footprint as a circle and assigns goal positions instead. This demonstrates that that we have improved over applying [1] to non-circular footprints. The plot also shows that the best coverage is achieved by selecting $\theta^*_{i,j}$ to minimize the Dubins distance from the current pose.

the semi-arbitrary paths returned by the goal planner when it fails to reach the goal). As expected, assigning angle-specific goals slows down goal assignment because the space of possible goals is much larger, and goal planning because the states consistent with a goal are much more restricted.

Here we can see clearly how screening for $\theta^*_{i,j}$ to minimize Dubins distance succeeds in picking much more reachable goals: it is much faster and succeeds much more often. The fact that this results in distinctly better coverage performance than the versions which select $\theta^*_{i,j}$ to optimize coverage urgency is somewhat surprising, but makes sense in light of how frequently the latter versions fail to find any path to the goal. Even more surprising is the fact that the paths found in the easiest-angle version average less than half the duration of those in the circle-approximation version.

VI. Conclusion

We present a method for assigning angle-specific goal poses in a persistent coverage scenario which balances the feasibility of reaching the goal with the priority of what it covers. Our proposed method provides significantly better coverage for angle-dependent sensor footprints than are achieved by approximating those footprints as if they are angle-independent, demonstrating that we have successfully generalized [1].

We also find that Dubins distance can improve the goal assigner's notion of how feasible it is to reach a pose, particularly when used to aggressively screen for only the most reachable angles at each potential goal location.

It was surprising that screening candidate angles for the most urgent coverage did not lead to better coverage results. Even when using feasibility as a factor in the cost function the Dijkstra search minimizes, this has a strikingly high rate of planner failure. After future work which may improve the goal planner, it may be worth re-examining the best way to select the candidate angle to associate with each cell.

When candidate angles are screened to minimize Dubins distance, the planner succeeds nearly as frequently as when it does not deal with angle-specific goals, and the resulting paths are dramatically shorter. The shorter paths to goals are, the more of the coverage is deliberate (selected with urgency in mind) rather than incidental (happening to be along the way to the goal). This dynamic suggests that future work integrating coverage priority directly into goal planning may be fruitful.

Other areas for future work could include replacing goal positions with goal regions (clusturs of nearby similar-urgency cells), learning parameters such as t_{max}, W_t, and r_{min} from data, and more sophisticated approaches to collaboration between UAVs.

Version	Average GA Time (s)	Average GP Time (s)	GP Failure Rate
Easiest Angle to Reach	0.7	0.8	10%
Most Urgent Angle	1.4	2.9	40%
Ignoring Dubins	1.3	3.0	40%
Circle Approximation	0.5	0.3	0%

TABLE I: Data from 2-hour runs of each version of the system.

Version	Average Plan Duration (s)	Average $C(t)$	Average $E(t)$
Easiest Angle to Reach	18.8	63%	20%
Most Urgent Angle	42.0	68%	23%
Ignoring Dubins	44.2	68%	24%
Circle Approximation	40.6	81%	30%

TABLE II: Data from 2-hour runs of each version of the system.

References

[1] T. Kusnur, S. Mukherjee, D. M. Saxena, T. Fukami, T. Koyama, O. Salzman, and M. Likhachev, "A planning framework for persistent, multi-UAV coverage with global deconfliction," in *In Proceedings of Field and Service Robotics 2019 (FSR19)*, 2019.

[2] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957. [Online]. Available: http://www.jstor.org/stable/2372560

Detecting Physiological State Changes During Blood Loss via Deep Unsupervised Learning

Chufan Gao[1], Anthony Wertz[1] and Artur Dubrawski[1]

***Abstract*— Monitoring of physiological responses during a blood loss event is crucial in determining appropriate treatment for the well-being of the patient. Physician intuition implies that the body has a number of different physiological response patterns to blood loss, which change as time passes and as blood loss worsens. Although previous research has shown that a random forest classifier is able to determine whether a patient is bleeding based on data alone, it is unclear whether a model is able to detect these accompanying response patterns from raw physiological data. To approach this problem, we use unsupervised machine learning techniques, such as K-means and Agglomerative clustering, as they are designed to extract patterns from data without a ground truth. However, since the data gathered from the patient are high-dimensional and in time series form, it is impractical to handle without further preprocessing. To make this tractable, we employ a deep dilated convolutional encoder with combined with a custom triplet loss function to project the data into a lower dimensional space. By clustering these latent vectors with time constraints and visualizing the clusters over time, we hypothesize that the clusters will correspond to the physiological response patterns that match physician intuition.**

I. Introduction

Internal bleeding is a common symptom from physical traumas, but it is difficult to analyze due to its complexity. The raw data produced by the monitoring equipment is often of multivariate time series form, which is high dimensional and difficult to visually analyze. Machine learning is a natural way of analyzing this high-dimensional data. However, as a whole, the internal bleeding process has not yet been extensively analyzed by the field of machine learning.

This is not to say that no work has been done–for example, Li et al. showed that prediction of whether a crash will happen during blood loss is possible [1]. Falck et al. found that for hemmorhage prediction, a GRU-based model achieves best performance in small false-positive range, while being inferior for negatives compared to a formidable baseline using manually extracted features and a random forest classifier [2]. Lei et al proposed a method of performing supervised classification canonical correlation clusters on time windows of CVP on a pig bleed dataset and was shown to perform well in prediction of bleeding vs non bleeding [3] (the appendix also has a list of medical terms and abbreviations for the reader's convenience). The work previously done on this data has focused on clear-cut, supervised or semi-supervised approaches in validating their hypothesis.

Previous work uses supervised learning to predict clearly delineated outcomes, like the mortality of a person in a given time window. However, there has been a lack of work focusing on less obvious changes in physiological data associated with blood loss. Supervised learning techniques are not applicable to discover these pattern due to the lack of ground truth in analyzing physiological state changes. Only a few studies use unsupervised learning to examine unlabeled data and discover important patterns. Utilizing an unstructured approach may allow us to more extensively understand different physiological effects of blood loss.

Furthermore, Lei et al's work demonstrated that interesting patterns could be found from the clusters - one cluster corresponded mostly to the prebleed phase, a second cluster would take over after bleeding started, and a third cluster would appear even further throughout the bleed. The interpretation was that the physiological responses reflected in the CVP data of the pigs were changing throughout the bleed and that these changes were different physiological responses. Their interpretation was that initially there is an initial compensation reaction to blood loss, and this quickly shifts into a secondary overall systemic reaction. Additionally, they found that most pigs that they analyzed all similarly exhibited such behavior.

This previous work serves as inspiration to our work, as it leads to many additional question concerning the nature of the physiological reactions. For example, are all physiological compensation events universal for each pig? How many such physiological responses are there? Can we build a model to learn and detect these responses?

Despite the widespread adoption of neural networks in data processing, continuous, multivariate time series data have not been affected insofar as, for example, the computer vision or natural language processing communities. The breadth of work in these fields has not translated to this multivariate time series data, even though these fields have valuable models that could be applied. Although deep unsupervised sequence processing techniques have generally focused on natural language processing, many of these model architectures can be generalized to continuous, multi-variable time series data with some additional effort.

[1]Chufan Gao, Anthony Wertz and Artur Dubrawski are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA chufang@andrew.cmu.edu; awertz@cmu.edu; jdolan@awd@cs.cmu.edu

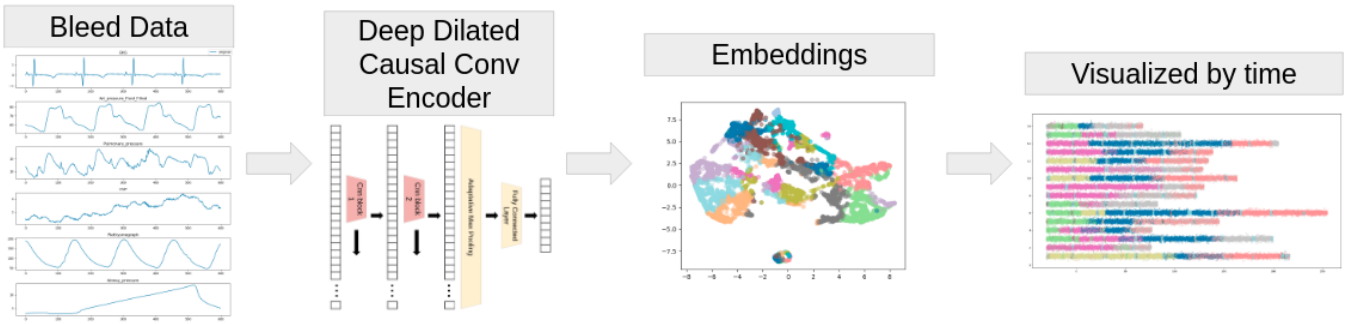


Fig. 1: Overview of the methodology

In this paper, we demonstrate a modern deep unsupervised encoder model in the application of finding embeddings from continuous data of 6 health metrics. With these embeddings, we can then use several different clustering techniques to obtain a number of clusters which may correspond to physician intuition. Additionally, this would expand on Lei et al's findings of physiological response patterns associated with blood loss.

A. Contributions

To summarize, the main contributions of this paper are

1) An investigation as to whether pigs in general have a universal physiological response pattern to internal bleeding.
2) An investigation as to the number of such physiological responses detected from the model.

II. Methodology

An overview is shown in 1.

A. Data

The data consists of health metrics of 93 pigs in total. These pigs are separated into 4 groups, which are then bleed at different rates - 60mL/min, 20mL/min, 5mL/min, and 0mL/min respectively. Each pig was monitored for 11 vital signs at 250 Hz (synchronized): arterial and venous blood pressures (CVP, arterial pressure fluid filled and millar, pulmonary pressure), arterial and venous oxygen saturations (SpO2, SvO2), EKG, Plethysmograph, CCO, stroke volume variation (Vigeleo), and airway pressure. The data collection methodology is similar to [4].

For our task, we choose to only use the 16 pigs assigned to the most gradual bleeding task - 5mL/min - as this should give us clearest indications of physiological responses as the pigs' status slowly worsens. Also, we choose to only use 6 of these features - EKG, arterial pressure fluid filled, pulmonary pressure, CVP, plethysmograph, and airway pressure - as they contain potential important semantic information about the physiological status. Additionally, we also have physician annotated timestamps and notes of when a blood draw is performed. This is important as performing a blood draw corresponds with extremely high variation noise in some variables in the time series for a few seconds. Each health metric is measured in 250 hertz.

Since we have 16 pigs and we want to simply analyze all of them, we train our encoder all 16 pigs. This ensures that we are training over all of the data and learning as much from the data as possible. For our training data, we pass in the entire bleed sequence of each pig to the model. To obtain embeddings of the bleed sequences, We choose a window of 600 timesteps and split from the time sequences (without overlap) time windows of 600 timesteps by 6 features. This allows us to get an embedding of the pig state for every 2.4 second window. We chose 600 as it was long enough to get 1 to 2 breaths in and was easier to process computationally; however, this is also a parameter that may be tuned in further research.

B. Causal Dilated Convolutional Neural Network

A convolutional neural network (CNN) is a neural network that trains well on even very high dimensional data, such as images. First introduced in 2012 [5], CNNs have long since been the cornerstone of modern computer vision approaches. However, CNNs can also be applied to non image data as well. For example, although recurrent neural networks have historically been used in sequential data modeling, CNNs are now a popular and viable approach in dealing with sequential data. Wavenet is one of the most famous example of this approach–audio waves produced by the CNN were better than previous state-of-the-art recurrent neural network approaches [6]. Wavenet used dilated convolutions, which skips a timestep every layer of the CNN - leading to an exponentially large effective view of the sequence. It also used causal convolutions, or the idea that the CNN can only process what it has seen before in the time sequence. In our application, causal convolutions are still important because it allows the model to be run on arbitrary length sequences and also be able to run on sequences online during test time. Should we extend our approach to online detection of bleeding responses phases, our model wouldn't break.

For our deep unsupervised embedding model we use an convolutional encoder as proposed by Franceschi et

al. [7]. Compared to an autoencoder, training solely an encoder model is beneficial in that it severely reduces the training time necessary for the model to achieve good performance; additionally, in many cases, training a decoder model is unnecessary to obtain meaningful embeddings, as shown by Franceschi et al. This model is able to produce meaningful embeddings that perform close to state of the art if not better compared to even supervised time series embedding methods [7]. Using CNNs to process sequences have also another inherent advantage: the speed. Since CNN operations are highly parallelizable, training this encoder is faster than training a traditional sequence to sequence recurrent neural network [7]. We choose to use this model as opposed to traditional statistical feature extraction as we want to assume as little as possible about the data and see if the network can discover patterns by itself. Further information about the nature of this convolutional structure is shown in Figure 2.

While only 3 dilated causal CNN layers shown here, in the actual model, the number of these modules is a hyperparameter that we can specify. The dilated part is visualized by the doubling of the gaps between the boxes that is analyzed by the CNN as you go further up in the output layers. The causal part is shown by the fact that the final output box on the upper right only has access to the information of the boxes before it–a regular CNN would appear more symmetric in that it have access to information after its current timestep as well. The adaptive pooling layer comes after the dilated causal CNN layers and simply reduces an arbitrary dimensional input to a fixed-sized output. Finally, the final layer is a fully connected layer that outputs the embedding.

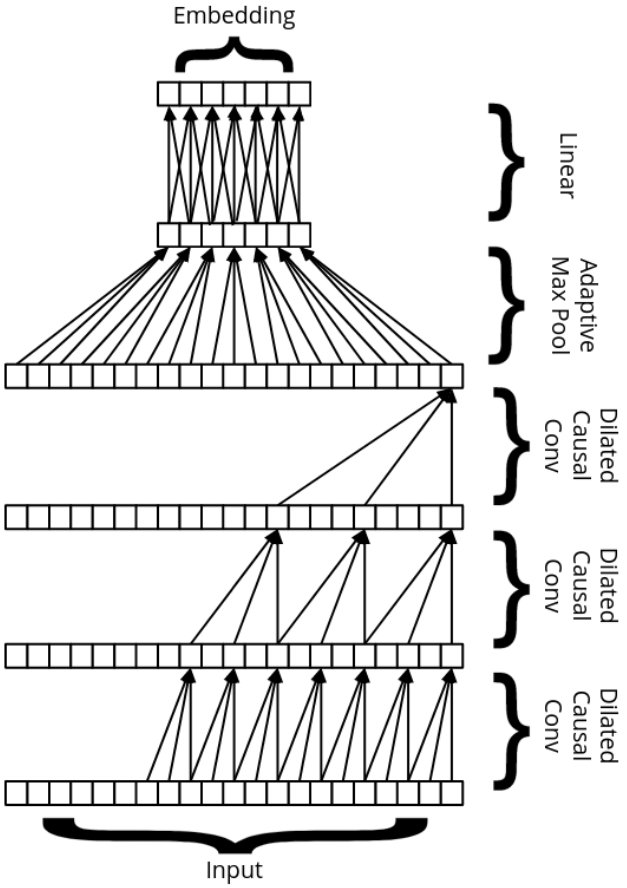


Fig. 2: Graphical representation of dilated and causal convolutions. The output of each row is the row above it. The bottom-most row of boxes denotes a variable length input, and the top-most row of boxes an out of latent vector embeddings.

C. Triplet Loss

We will use triplet loss to train our encoder as specified by [7] et al. Triplet loss is a loss function with a very natural intuition as its basis - similar things should be close together and unsimilar things should be further apart. This is reflected in its mathematical formulation. Let f be our encoder that obtains latent vectors from the time series data. Let x, x^{pos}, x_k^{neg} be the reference time series, a positive time series example, and a negative time series example. Let K be the number of negative samples to take. Then, the loss is shown in equation 1.

$$L = -log(\sigma(f(x)^T f(x^{pos}))) - \sum_{k=1}^{K} log(\sigma(-f(x)^T f(x_k^{neg}))) \quad (1)$$

Triplet loss is popular in natural language processing - Word2vec [8] as it is effective in training unsupervised models that obtain latent vectors from words that encode some semantic meaning. Franceschi et al. demonstrated that this is useful for unsupervised learning of useful embeddings of general multivariate time sequences as well [7].

D. Sampling methodology

We use a modified version of Franceschi et al.'s sampling algorithm to obtain choices of reference x, positive example x^{pos}, and negative example x_k^{neg}. This is different from the original implementation from Franceschi et al. since the negative samples are only choosen randomly when there can be no overlap with the reference time series; thus, this guarantees that a negative example can't be a positive example as well. This should allow the model to learn better as there is a clearer difference between positive and negative samples. Algorithm 1 shows the methodology. Figure 3 shows how our proposed sampling algorithm in practice.

E. Evaluation

Since we have no ground truth, validation of usefulness of these embeddings is an open problem. However, we can qualitatively evaluate them. We use a variety of different clustering methods to try to find the one that produces the best cluster of embeddings that makes sense to us intuitively. We are looking for is separation of the clusters by time - that is - different clusters should be separated

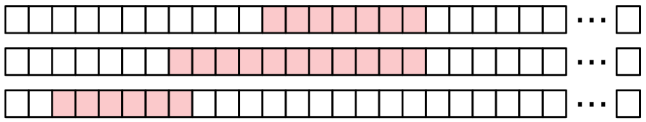


Fig. 3: Sampling methodology. Let the each row of boxes represent multivariate sequences of our training data, and let the rows be the sequences in the batch where we choose the references from. Then, the pink squares represent the reference timesteps, positive timesteps are sampled form within the reference timesteps. Negative timesteps are sample from the white boxes (all of the timesteps excluding the reference timesteps).

for $i \in [1, N]$ **do**
  randomly choose length of reference sample $len_i^{ref} \in [1, len(y_i)]$
  randomly choose length of positive sample $len_i^{pos} \in [1, len_i^{ref}]$
  randomly choose reference sample x_i^{ref} from subseries of y_i of length len_i^{ref}
  randomly choose positive sample x_i^{pos} from subseries of x_i^{ref} of length len_i^{pos}
end for
for $k \in [1, K * N]$ **do**
  randomly choose y_k from the batch
  randomly choose $len_k^{neg} \in [1, size(y_k)]$
  Let $x_{y_k}^{ref}$ be the reference sample that we previously took from y_k
  randomly choose x_k^{neg} among subseries of y_k of length len_k^{neg} without overlapping $x_{y_k}^{ref}$
end for

Algorithm 1: Input: A training batch of complete sequences y_i, where i is the ith training sequence. Let N be the total number of sequences in this training batch. Let K be the ratio of negative samples to be sampled per batch item. Output: N reference samples x^{ref}, N samples of positive samples x^{pos}, and N*K negative samples x^{neg}.

from each other by time, but the same cluster should not be separated by time. Another thing that we are looking at is order consistency of the clusters over time. For example, since the pigs are only ever getting worse in our data, and should never return to a previous "healthy" state.

Specifically, we use a number of different time embeddings, clustering techniques, and latent dimensions, and finally, number of clusters. We explore:

1) Time embeddings added to latent embeddings. The type of time embedding is taken from attention transfomers, from Vaswani et al. [9].
   a) No time information added
   b) Adding time information for full length of the sequence
   c) Adding time information only from the start of bleed (Since we know the exact location of the start of bleed from the physician annotations, we only add temporal information to the embeddings obtained after the pig starts bleeding. The prebleed embeddings are left alone. This is effectively adding information about the amount of blood lost, since bleed speed is constant after the pig starts bleeding).
2) Clustering methods (All of these are implemented in sklearn [10])
   a) K-means
   b) Agglomerative clustering with ward linkage (Bottom-up hierarchical clustering. Ward's linkage merges the two clusters such that the increase in the value of the sum-of-squares variance is minimized [11]. Specifically, sklearn references [12].)
3) Latent embedding dimensions: 64, 128, 256
4) Number of clusters: In addition to all of these methods, we also explore 11 different numbers of clusters that we pass into the clustering algorithms (from 2 to 12 clusters).

III. Results

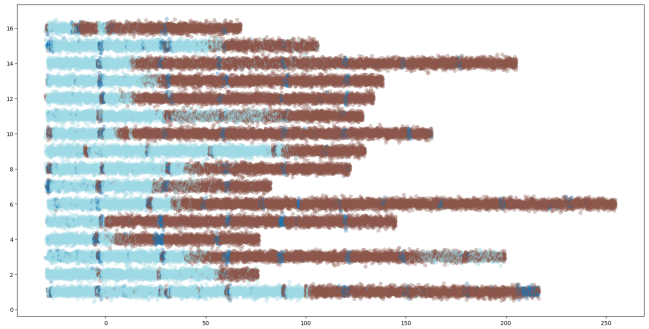
A. Graphs of the clusters

Since there may be 3 clustering methods * 3 time embedding methods * 3 latent embedding dimensions * 11 number of clusters = 297 possible graphs in total, it is impractical to show all of them in this paper. Thus, we only show a few examples shown in Figure 4. However, all of the figures and model code (along with hyperparameters) will be on my github: github.com/andy1445. The Y axis of each of the following figures represents each pig, and the X axis represents minutes. The different colors represent different clusters found by the clustering algorithm, and the colors are random. To make these plots easier to see, we jittered the Y axis.

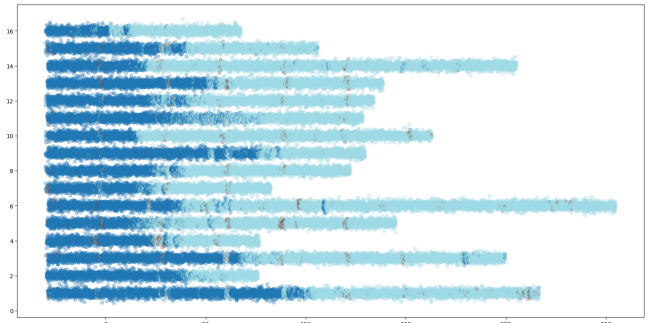
IV. Discussion

From these graphs, we can see that the clustering algorithm is at able to discern a "healthy" state and a "non healthy" state for all of the 3 cluster graphs. For the 11 cluster graphs, for both kmeans and agglomerative clustering we are able to discern between different pigs as well as different cluster progressions throughout the bleed; however, all pigs eventually end up in the same state - the green or purple cluster for kmeans and the orange cluster for agglomerative clustering. This make sense as the pigs all crash after this last cluster. Additionally, we found that reactions between pigs are not universal, as some pigs skip clusters entirely through the bleed, and they can also start off in different clusters compared to other pigs. We also see that pigs can go through as many as 5 different states and as low as 2 states. Additionally, we are able to detect bleed draws. They show up as noise that occurs regularly every few 30 minutes or so in the plot of the latent clusters.

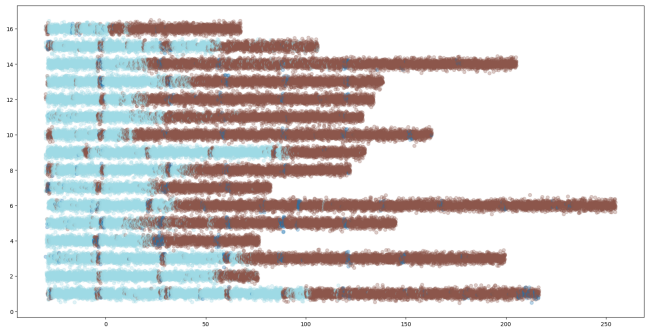
Fig. 4: The plots of the clusters over time when the model is trained with the ability to differentiate between differences between individual pigs. The colors of the clusters are not consistent between the plots. The X axis is in minutes, and the Y axis are the pigids.



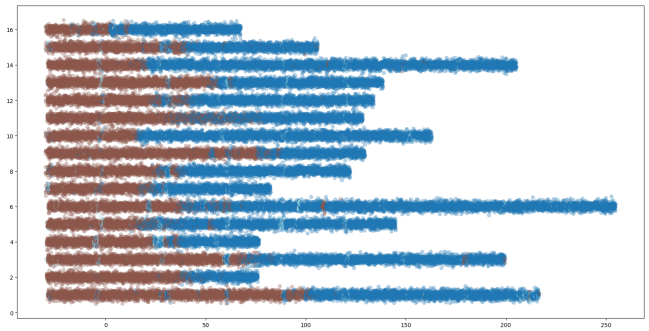
(a) 128 dimensional latent vectors, half time embedding, agglomerative clustering, 3 clusters



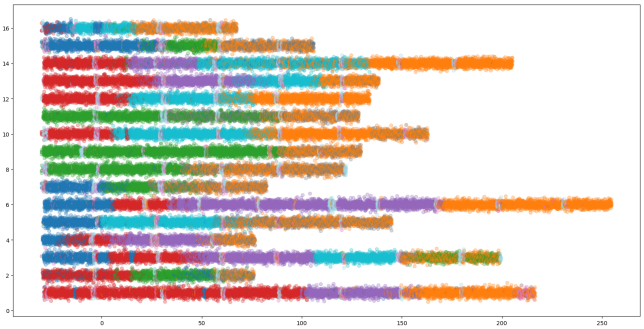
(b) 128 dimensional latent vectors, half time embedding, kmeans clustering, 3 clusters



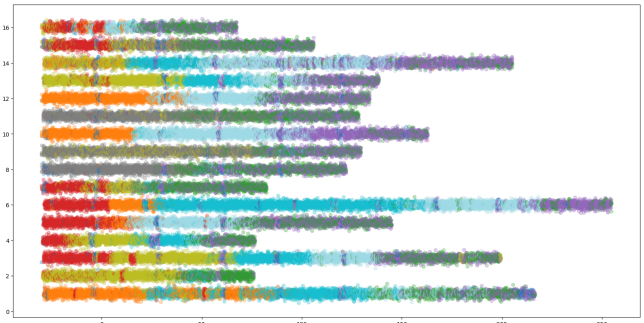
(c) 128 dimensional latent vectors, no time embedding, agglomerative clustering, 3 clusters



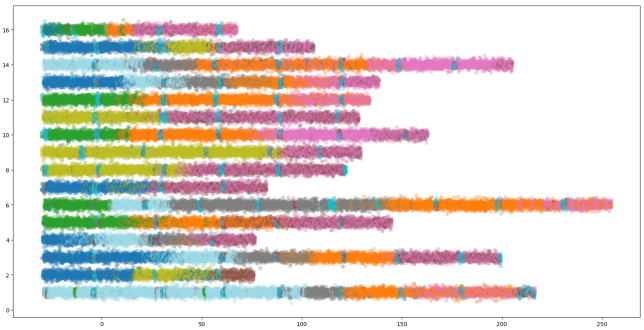
(d) 128 dimensional latent vectors, no time embedding, kmeans clustering, 3 clusters



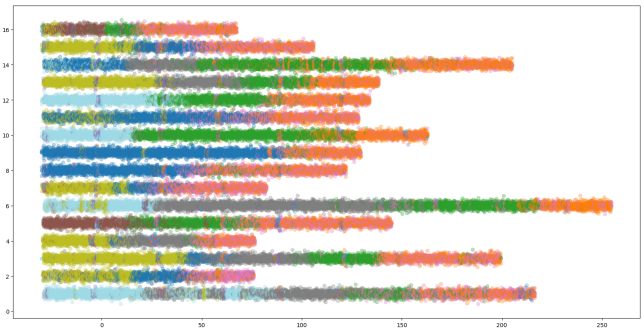
(e) 128 dimensional latent vectors, half time embedding, agglomerative clustering, 11 clusters



(f) 128 dimensional latent vectors, half time embedding, kmeans clustering, 11 clusters



(g) 128 dimensional latent vectors, no time embedding, agglomerative clustering, 11 clusters



(h) 128 dimensional latent vectors, no time embedding, kmeans clustering, 11 clusters

V. Future Work

Future work should seek to use more rigorous evaluation metrics. Once we have the labels, we can then train a random forest classifier to predict the labels from the raw embeddings alone. Additionally, we should also use our embeddings to predict bleed or survival, like previous work has done. Future work should also explore more variety of models such as different encoder models such as Variational Autoencoders. Additionally, Additionally, we can also find clusters or separations in a multitude of different ways, for example, we can use Hidden Markov Models (HMMs) to detect changes as well as change point detection. We should also aim to have physicians analyze the validity of the different clusters that we find.

VI. Acknowledgements

The authors would like to acknowledge the Robotics Institute Summer Scholar program for facilitating this research. We'd also like to acknowledge the Auton Lab, for the invaluable guidance from its members throughout the summer.

Appendix

Abbreviation and Definitions

- Airway pressure: Pressure in the the airways, note that this is under artificial ventilation to offset the energy requirement for breathing, so that a cleaner reaction to the blood loss may be obtained (measured in mmHg) [2].
- Arterial pressure fluid filled: Systemic arterial blood pressure in the aorta, measure in mmHg [2].
- Arterial pressure millar: Systemic arterial oxygenated blood pressure in the peripheral (measured in mmHg) [2].
- CCO: Continuous cardiac output is a measure of the volume of blood pumped from the heart in a certain amount of time. It is often used as a predictor of oxygen delivery to the cells (measured in mL/s) [13]
- CVP: Central venous pressure is a measure of pressure in the superior vena cava that can be used as an estimation of right atrial pressure, often used as an assessment of hemodynamics and hemmorage prediction, particularly in intensive care units (measured in mmHg) [14].
- EKG/ECG: Both are the exactly the same thing and stand for electrocardiogram, which is a measure of the flow of the cardiac electrical cycle (measured in mV) [15].
- Plethysmograph: A waveform that represents changes in blood volume. It has no units and is qualitative due to the non-linear relationship between the absorption of the light for each individual, but overall patterns can be [16].
- Pulmonary pressure: Pulmonary artery pressure (PAP). The pressure of blood pumped from heart into pulmonary (lung) system (measured in mmHg) (i.e. deoxygenated) [2].
- SpO_2: Arterial oxygen saturation (oxygenated) (measured in %) [2].
- SvO_2: Venous oxygen saturation (deoxygenated), (measured in %) [2].
- Vigeleo: Variation of the stroke volume (SV) or the volume of blood pumped out of the left ventricle (oxygenated), defined as range over mean [2].

References

[1] X. Li, M. R. Pinsky, G. Clermont, and A. Dubrawski, "Leveraging routine pre-operative blood draws to predict hemorrhagic shock during surgery," in *Proceedings of the Neural Information Processing Systems Machine Learning for Health (ML4H) Workshop 2018*, 2018.

[2] F. Falck, M. R. Pinsky, and A. Dubrawski, "Deep sequence modeling for hemorrhage diagnosis," in *Proceedings of the Neural Information Processing Systems Machine Learning for Health (ML4H) Workshop 2018*, 2018.

[3] E. Lei, K. Miller, and A. Dubrawski, "Learning mixtures of multi-output regression models by correlation clustering for multi-view data," *arXiv preprint arXiv:1709.05602*, 2017.

[4] M. R. Pinsky, "Instantaneous venous return curves in an intact canine preparation," *Journal of Applied Physiology*, vol. 56, no. 3, pp. 765–771, 1984.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[7] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *arXiv preprint arXiv:1901.10738*, 2019.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] T. W. Liao, "Clustering of time series dataa survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[12] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint arXiv:1109.2378*, 2011.

[13] J.-L. Vincent, "Understanding cardiac output," *Critical care*, vol. 12, no. 4, p. 174, 2008.

[14] P. Shah and M. A. Louis, "Physiology, central venous pressure," in *StatPearls [Internet]*. StatPearls Publishing, 2018.

[15] M. B. Conover, *Understanding electrocardiography*. Elsevier Health Sciences, 2002.

[16] J. Nijboer, J. Dorlas, and H. Mahieu, "Photoelectric plethysmography-some fundamental aspects of the reflection and transmission methods," *Clinical Physics and Physiological Measurement*, vol. 2, no. 3, p. 205, 1981.

Anomalous Pattern and Systemic Error Detection in Radiation Portal Monitors via Robust Deep Autoencoders

Harshayu Girase[1], Kyle Miller[2] and Artur Dubrawski[2]

***Abstract*— Detecting anomalous points and patterns has always been a highly researched topic in the machine learning community. Our work focuses on detecting anomalous patterns and potential systemic malfunctions in radiation portal monitors (RPMs). Since RPMs have over hundreds of sensors that collect real-time data, it is not uncommon to expect systemic malfunctions due to sensor bias and noise from time to time. Most RPMs are often unaffected by minor malfunctions; however, it is of critical importance to detect these slight errors, as there is always a risk of missing hazardous radioactive materials. Prior research on this topic involves distance and non-parametric based detection; however, the performance of these methods often does not scale well in high-dimensional spaces. Thus, more recent literature focuses on PCA and spectral clustering based methods to detect unusual patterns. To leverage complex relationships present in high-dimensional data, we use state-of-the-art robust deep autoencoders [1], [2] along with a changepoint detection algorithm to detect anomalous changes in data. Our method detects periods of anomalous patterns and small sensor malfunctions, for example bias and unexpected noise, with higher accuracy on more subtle errors.**

I. Introduction

Following the attack on 9/11, US Customs and Border Protection deployed thousands of RPMs across national borders to detect radiological threats. One of the most comprehensive methods used for radiation detection is a tool developed by Lawrence Livermore National Laboratory: Algorithm Development for the Enhanced Radiological Nuclear Inspection and Evaluation (ERNIE). ERNIE classifies passing vehicles into one of the following 6 categories NonEmitting, NORM, Medical, Industrial, Fissile, and Contamination based on materials present in the vehicle. ERNIE extracts various features via vehicle motion profiling and classifies radioactive materials using machine learning and algorithmic based classifiers [3].

While there has been significant R&D into building robust classifiers, RPMs suffer from sensor drift and noise introduced from external factors. To combat potential systemic issues, port staff often look out for unusual sensor behavior and calibrate sensors with noticeable drift or noisy outputs. However, there are many issues with human-based detection — most notably that RPMs can have sensor malfunctions without a significant effect on output, leading them to go unnoticed.

[1]H. Girase is a junior student in Department of Computer Science, University of California, Berkeley harshayugirase@berkeley.edu

[2]Kyle Miller & Artur Dubrawski are with the Auton Lab, Robotics Institute, Carnegie Mellon University, USA mille856@andrew.cmu.edu, awd@cs.cmu.edu

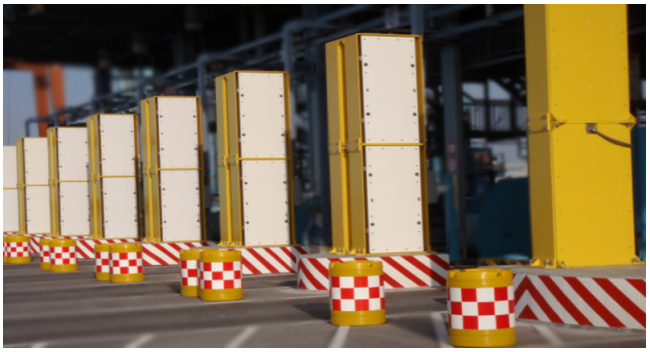


Fig. 1. An image of a radiation portal monitors at US border

Hawkins [4] defines anomalies as observations that deviate so significantly from other observations as to arouse suspicion that it was generated by a different mechanism. Much prior work in anomaly detection relies on non-parametric distance and density based models, which tend to scale poorly as data becomes higher dimensional [5]. Many papers on anomaly detection for radiation data have also explored PCA and RPCA based anomaly detection [6], but there are several limitations with these methods. With the rise of deep learning, there have been many breakthroughs in deep-learning based anomaly detection. Deep networks can leverage complex relationships present in this highly multi-dimensional setting to identify anomalous data that may be generated by a different process than what is expected. This paper focuses on identifying anomalous patterns in data rather than singular anomalies. Through this detection, we aim to create a more robust system that will notify port staff about potential issues in RPMs that may have otherwise gone unnoticed.

II. Relation to prior work

Detecting anomalies in radiation data is a tough task due to the high-dimensional nature of RPM data. Beyer et. al [5] shows the implications when the ratio of the variance of the length of a vector ($x \in \mathbb{R}^d$) in feature space to the length of the mean vector ($\mathbb{E}[x] \in \mathbb{R}^d$) in feature space approaches 0 as the dimensionality increases:

$$\lim_{d\to\infty} var\left(\frac{\|x\|}{\|\mathbb{E}[x]\|}\right) = 0 \implies \frac{D_{max} - D_{min}}{D_{min}} \to 0 \quad (1)$$

From equation (1) we see that the proportional distance between the closest and farthest point vanishes with increasing

dimensions, posing issues for common density and distance based outlier algorithms. Below we outline several current methodologies used for anomaly detection in this field and address pitfalls in each of these techniques.

A. PCA and Robust PCA

The Mahalanobis distance metric is quite popular when it comes to anomaly detection; however, its effectiveness reduces due to the hundreds of sensors present in RPMs. To combat this, many approaches [6], [7] have performed Mahalanobis-based thresholding following PCA dimensionality reduction. These papers rely on the fact that the top principle components capture majority of the data's variance. Thus, they project the data onto a linear, lower dimensional subspace and then perform anomaly detection using Mahalanobis distance as a metric for how "anomalous" data may be.

$$RootMD = \sqrt{(x-\bar{x})^T \cdot S^{-1} \cdot (x-\bar{x})} \tag{2}$$

Both [6] and [7] use equation (2), where x is the projected data point, $\bar{x}$ is the mean of all projected points, and S^{-1} is the inverse covariance matrix of the data, to attribute a Mahalanobis-based score.

While Mahalanobis distance would work better in this lower dimensional space, there are a few fundamental issues with PCA based anomaly detection. Chalapathy et. al discuss the two major limitations of PCA. First, it is highly sensitive to perturbations in data — that is few unexpected training points will result in a completely different lower dimensional representation of the data. Second, it forces the data to be captured effectively in a linear subspace. While the first issue has been addressed through robust PCA, the second is still unanswered [1]. Furthermore, in our experiments we still needed more than 50 dimensions to avoid significant information loss, which in some scenarios would still pose problems for distance-based clustering algorithms.

B. Spectral Clustering

There has been work done to leverage local relation between points via spectral clustering. The main claim as to why PCA and clustering techniques fail is that they only perform well in cases where normal data falls into a dense region and anomalous data falls into sparser regions. Furthermore, the magnitude of features and distance thresholds are challenging parameters to tune [8]. The authors of [8] find lower dimensional projections using spectral methods without assumptions about data density. They claim this method works better than PCA, as principle components sometimes fail to capture important local patterns unlike spectral clustering. As discussed later, we noticed that subtle anomalous patterns are hard to notice even with spectral methods such as TSNE [9].

C. Our Approach

With large amounts of data, deep neural networks tend to perform better than traditional machine learning methods. This is primarily due to the fact that deep models can capture complex structure in data and can learn hierarchical discriminative features from the data to detect anomalies [10].

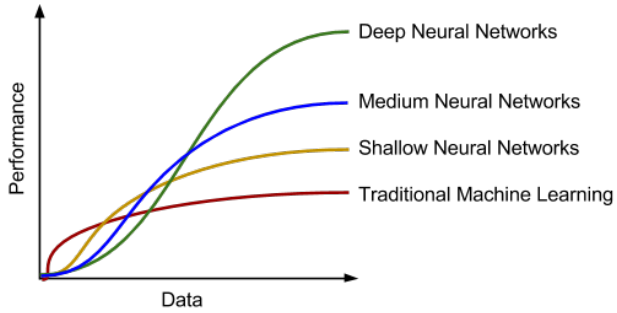


Fig. 2. Scaled Performance of Neural Networks vs. Traditional Models [11]

Autoencoders, a popular deep neural network model, have been shown to detect subtle anomalies better than methods such as linear PCA and kernel PCA, as they incorporate nonlinear dimensionality reductions [12], [13]. In addition, autoencoders can learn correlations between features, unlike PCA which projects features onto an orthogonal basis. However, autoencoders like traditional PCA, tend to be sensitive to highly noisy data and outliers. Thus, we explore a more robust scheme introduced in [2] of robust autoencoders. We train a deep robust autoencoder on our data (which inherently contains a few anomalies/noisy input points) and use reconstruction error of the test data as a metric to determine how well the testing data is represented by our training set. Significant changes in reconstruction error (i.e. sudden increases) may indicate that the new data may be generated differently than expected. We are more focused on detecting these changing patterns rather than singular anomalies. To flag significant changes in reconstruction error, we use a changepoint detection algorithm. To our knowledge, no paper has explored the use of robust autoencoders and changepoint detection to identify anomalous patterns in radiation data.

III. Methodology

A. Anomaly Detection via Autoencoders

Autoencoders are neural networks that learn a nontrivial function to map a vector $x \in \mathbb{R}^n$ to a vector $\tilde{x} \in \mathbb{R}^n$ where the reconstructed vector, $\tilde{x} \approx x$.

In figure [3], we see two key steps in the network: encoding and decoding. The encoder is a function $E\colon X \rightarrow Z$ where $X \in \mathbb{R}^n$, $Z \in \mathbb{R}^m$ and $m << n$. The encoder transforms input points into a lower dimensional representation through nonlinear transformations such as ReLU and sigmoid functions. The decoder is a function $D\colon Z \rightarrow \tilde{X}$ where $\tilde{X} \in \mathbb{R}^n$. The decoder reconstructs the lower dimensional

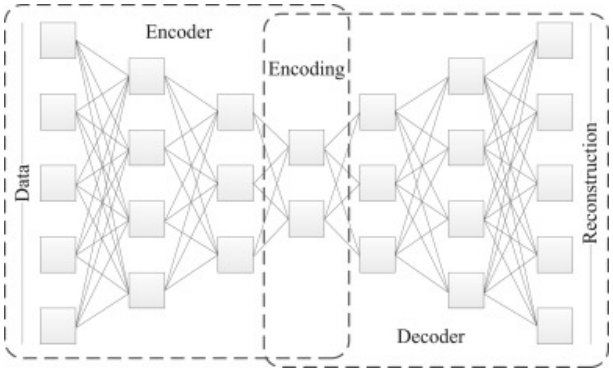


Fig. 3. Image of a 5-4-3-2-3-4-5 autoencoder [11]

embedding back to a point in the original space. The typical loss functions used for autoencoders are L_p norms or cosine-similarity. We use the following reconstruction loss function:

$$TotalMeanAbsoluteError = \frac{\sum_{x \in X} |x - D(E(x))|}{|X|} \quad (3)$$

We use mean absolute error rather than higher L_p norms, so as to penalize singular abnormalities in features less. In our experiments, MAE and MSE produced empirically similar results. In minimizing the MAE (3), the autoencoder learns a lower dimensional subspace via nonlinear transforms to capture the most important aspects of the data. Thus, in theory, the autoencoder will produce a low reconstruction error for normal data, as it has learned a well-represented lower dimensional embedding for the majority of data, and produce higher reconstruction errors for anomalous data, as the information of these points is not captured effectively in the lower dimensional embedding.

B. Robust Autoencoders

Similar to PCA, autoencoders suffer greatly from outliers present in training data, as it learns to represent these few anomalies in a lower dimensional space to prevent incurring high reconstruction penalties for these unusual points. In doing so, the latent representation learned by the autoencoder does not encapsulate the normal data as effectively. While training regular autoencoders, we noticed that the training set had abnormally high reconstruction error for 1% of data. We adopt the framework proposed in [2] to train robust deep autoencoders to learn a better latent representation. The proposed method splits the training data into two disjoint sets $X = L + S$ where $\{L\}$ ideally contains the normal data well represented by the latent encoding and $\{S\}$ contains outliers and noisy inputs. To balance minimizing reconstruction error and putting all points in set $\{S\}$, we minimize the following joint cost function [2]

$$\begin{aligned} \min_{\theta} \quad & \|L - D_\theta(E_\theta(L))\|_p + \lambda \cdot \|S\|_0 \\ \text{s.t.} \quad & X - L - S = 0 \end{aligned} \quad (4)$$

Here, λ is a tunable parameter to penalize the size of set $\{S\}$. As $\lambda \to \infty$, we simply get a regular autoencoder and as $\lambda \to 0$, all points will be placed in $\{S\}$ as noisy points and outliers, trivializing the optimization problem. Thus λ should be tuned based on how many points should be allowed in $\{S\}$. Currently, equation (4) is non-convex, making it intractable. Using similar motivations from RPCA, [2] relaxes the combinatorial $\|S\|_0$ term to $\|S\|_1$ making it convex and tractable:

$$\begin{aligned} \min_{\theta} \quad & \|L - D_\theta(E_\theta(L))\|_p + \lambda \cdot \|S\|_1 \\ \text{s.t.} \quad & X - L - S = 0 \end{aligned} \quad (5)$$

There are other regularization schemes for S such as $l_{1,2}$ regularization, but we use the standard l_1 penalty, and assume unstructured noise/outliers [2].

C. Changepoint Detection

After reconstructing unseen test data, we use a changepoint detection algorithm to identify significant changes in reconstruction error. If there is a significant increase in reconstruction error, we have reason to believe that there may have been some systemic malfunction that introduced this anomalous pattern; thus, we flag it for port staff to further investigate. For now, we use an offline detection framework from the ruptures changepoint detection library [14].

After collecting sequential data from RPMs and calculating reconstructing error for these points, we have a signal $R = \{r_1, r_2, ..., r_T\}$ We assume that if there are anomalous patterns in the data, perhaps introduced by sensor bias or noise, there will be abrupt shifts in the reconstruction signal. Let's assume there are K significant shifts in the signal at instances $t_0^* < t_1^* < ... < t_K^*$. We want to find the optimal K where $0 < K < T$. To find these shifts, we first define a total cost function [15]:

$$TotalCost = \sum_{i=1}^{K} \left[C\left(R_{(t_{i-1}:t_i)}\right) \right] + \beta \cdot \rho(K) \quad (6)$$

Here, $C(subsignal)$ is a cost function for a segment of the reconstruction signal. In our case, given a subsignal $Y = \{y_a, y_{a+1}, ..., y_b\}$ with median $\bar{y}$, the cost would be $\sum_{e \in Y} \|e - \bar{y}\|_1$. The term $\beta \cdot \rho(K)$ is a weighted penalty term to avoid overfitting and detecting $K = T$ changepoints. We used the binary segmentation method [14] to find an approximate solution to this optimization problem.

IV. Experimental Setup

A. Dataset

We use a dataset provided by Lawrence Livermore National Laboratory (LLNL) that contains both synthetic and real-world radiation data. The dataset contains sensor data for alerts that include NonEmitting, Normally Occuring Radioactive Materials (NORM), Medical, Industrial, Fissile, and Contaminated materials. We extract important features from this dataset, as it contains over hundreds of features.

Some extracted features include top PCA components, key features in the Fourier domain, vehicle data via motion profiling, and signal-related information such as peak distances. Even with extracting important features, we have over 150 features, which is still relatively high dimensional for current anomaly detection methods [6], [7], [8] to work effectively.

B. Simulating Anomalous Patterns

Sensor bias and noise is a common issue radiation portal monitors face. Early detection and calibration of these faulty sensors is critical to prevent further performance deterioration. Spontaneous sensor bias and noise are introduced in RPMs due to a variety of external factors, including physical imperfections of material, poor mechanical coupling and electrical issues, and random temporal phenomena of thermal and photonic origins [16]. One of the toughest tasks in the field of anomaly detection is developing a robust evaluation metric. This problem arises because ground truth anomalies often do not exist. We adopt a similar scheme as [17] to simulate potential system malfunctions. Some of the most common distributions associated with noise are Gaussian, Binomial, Poisson, Bose-Einstein, and Fermi-Dirac distributions [16]. In our simulations, we experiment with gaussian noise, as it is one of the most common and simple forms of noise.

Algorithm 1 Simulate Faulty Sensors

Initialize: probability failure, p: 0.05
Loop:
for $j \leftarrow 0$ to $len(features)$ **do**
  $rand = randomNumber(0, 100)$
  if rand $< p \cdot 100$ **then**
    bias = $10^{-4} \cdot min(Feature_j)$
    variance = $std(Feature_j)$
    error = $\mathcal{N}(bias, variance)$
    $Feature_j = Feature_j + error$
  end if
end for

We run algorithm 1 on the entire dataset and concatenate the normal and anomalous data. We now have 50% normal data and 50% anomalous data. We perform two tests of detecting these anomalous patterns. First, we simply test if our algorithm can detect a change from normal to anomalous by having just a singular changepoint. Second, we simulate periods of systemic malfunction by simply injecting 2 periods of anomalous data in the normal data.

C. Effect of Anomalies

After adding noise to roughly 5% of the features to simulate malfunctioning sensor readings, we noticed that classifier had an insignificant decrease in performance — that is, the classifier can handle these minor malfunctions fairly well. This makes it even harder to detect these errors, as the "anomalous" data can barely be discriminated from normal data.

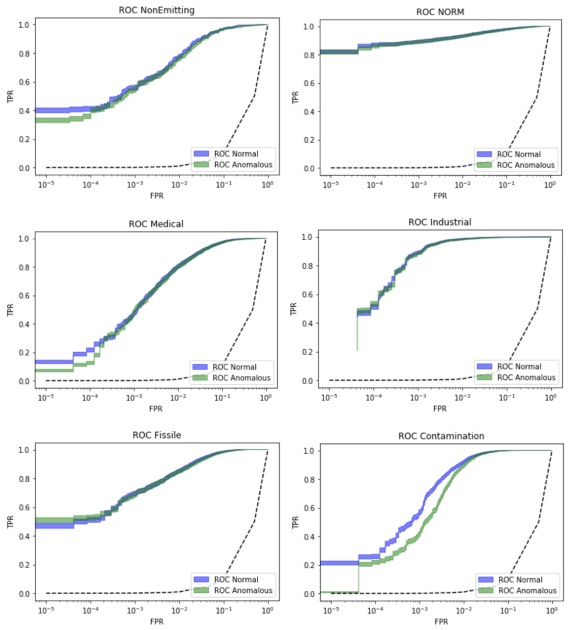


Fig. 4. ROC Curves of All Classes with 95% confidence interval (Log Scale). There is clearly an insignificant drop in performance even with data generated with faulty sensors. Thus, detecting sensor malfunctions purely based on performance drop may not work in case of minor errors.

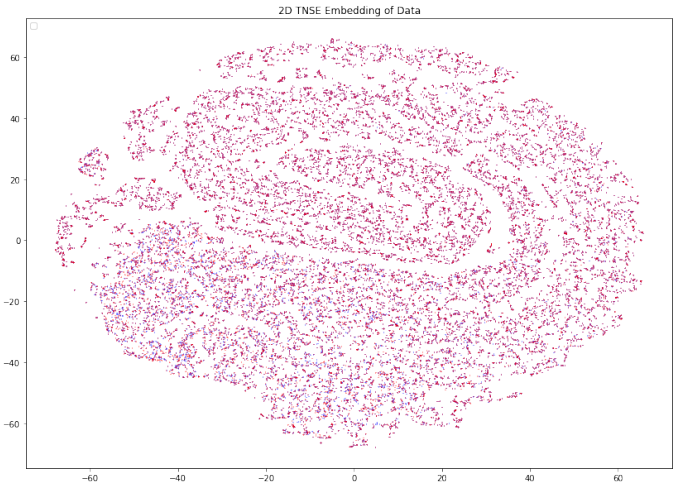


Fig. 5. TSNE Embedding of Data (Blue=Normal, Red=Anomalous)

Figure 4 shows that there is an insignificant drop in classification performance (we use a robust implementation of random forests). Classification accuracy drops from 91.8% to 91.4% which is unnoticeable and statistically insignificant.

V. Results

A. TNSE Projection

From figure 5, we can observe that there is no clear visual distinction between normal and anomalous data. There are a few small regions of densely clustered red points, but it would be hard to algorithmically tune parameters to determine significant changes between anomalous and normal data based on these embeddings. Because TSNE

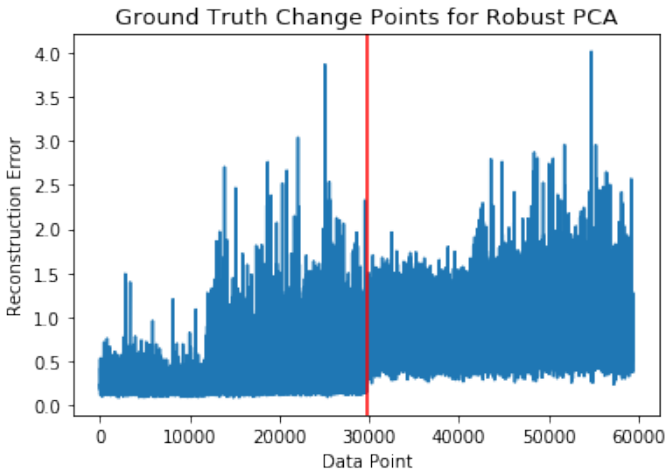


Fig. 6. Ground truth changepoint for RPCA (red line indicates changepoint from normal to anomalous). [Experiment 1]

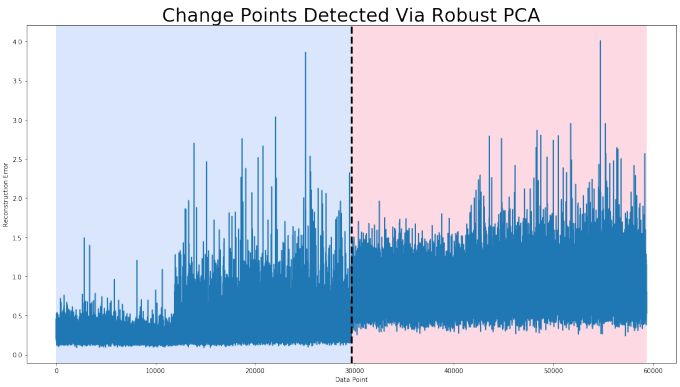


Fig. 7. PCA correctly detects changepoint via changepoint algorithm. [Experiment 1]

approximates spectral clustering [9], we can see that spectral clustering based approaches for anomalous pattern detection [8] may perform poorly when there are very minute malfunctions in RPMs.

B. Experiment 1, Simulating Singular Malfunction

All data is projected onto a 75 dimensional subspace for both autoencoders and PCA. In these figures, the first $\sim$ 30,000 points are normal data while the latter $\sim$ 30,000 points are those generated by a simulated malfunctions. A changepoint is defined when the data changes *normal* $\rightarrow$ *anomalous*.

1) RPCA Performance: From figures 6 and 7, we see that changepoint detection on RPCA-based point reconstruction can accurately identify changes from normal to anomalous data. An interesting observation here (figure 7) is that RPCA does not learn all of the normal data equally. Later data points ($\sim$12,000 to $\sim$30,000), which belong to different classes than earlier data points (0 to $\sim$12,000), have a noticeably higher reconstruction error than earlier data points. This imbalance in representing normal data based on class may raise false alarms simply based on changes due to class reconstruction.

2) Robust Autoencoder Performance: Figures 8 and 9 show that changepoint detection based on the autoencoder's reconstruction also correctly identifies the change from normal to anomalous behavior. However unlike the PCA reconstruction, the autoencoder learns the normal data more evenly (reconstruction error for normal data (points 0 to $\sim$30,000) in figure 9 is similar across all points).

C. Experiment 2, Simulating Multiple Malfunctions

All data is projected onto a 75 dimensional subspace for both autoencoders and PCA. In these figures, we have 3 changepoints: *normal* $\rightarrow$ *anomalous* $\rightarrow$ *normal* $\rightarrow$ *anomalous*.

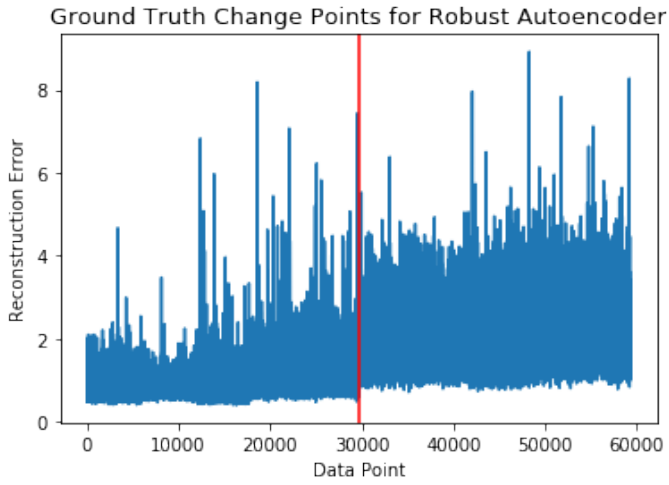


Fig. 8. Ground truth changepoint for autoencoder (red line indicates changepoint from normal to anomalous). [Experiment 1]

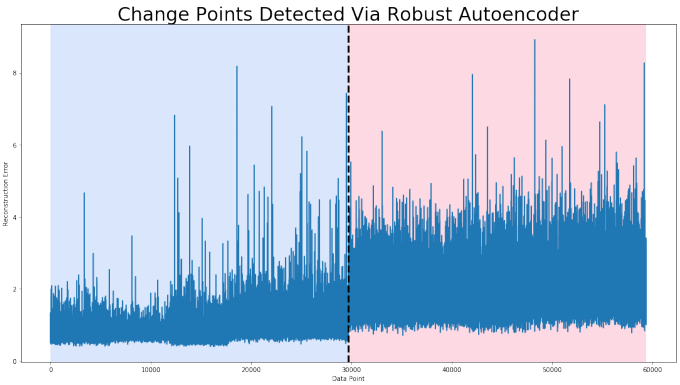


Fig. 9. Autoencoder correctly detects changepoint via changepoint algorithm. [Experiment 1]

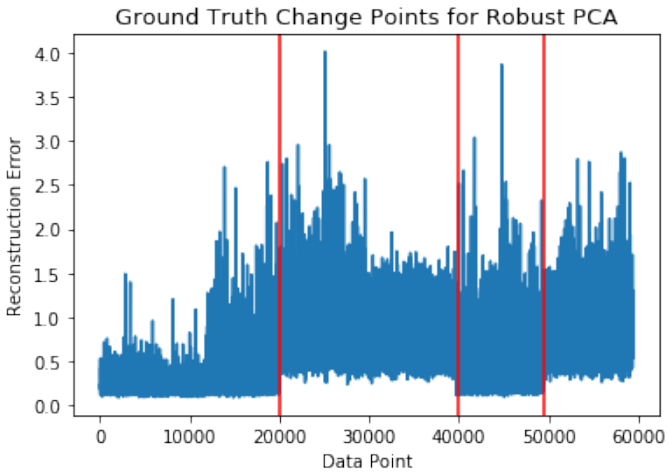


Fig. 10. Ground truth changepoints for RPCA (red line indicates changepoint from normal to anomalous or anomalous to normal). [Experiment 2]

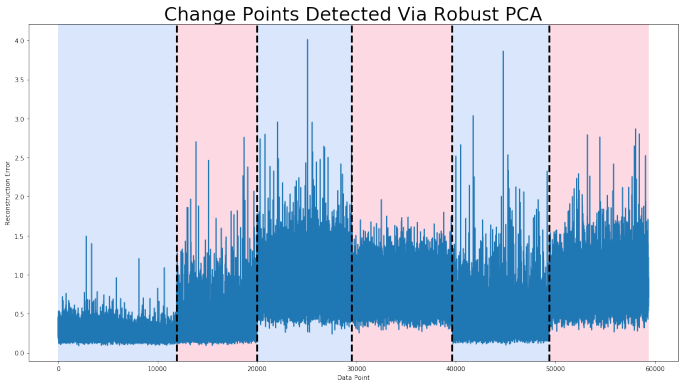


Fig. 11. RPCA correctly identifies 2 out of the 3 changepoints. However, it raises 3 false alarms due to imbalanced latent representation of different classes. [Experiment 2]

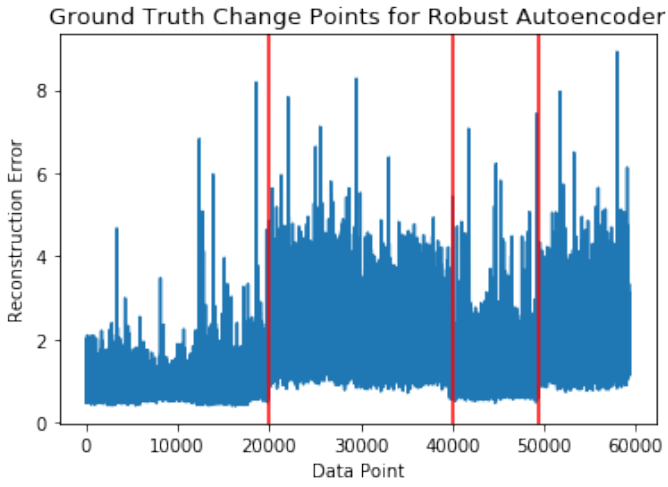


Fig. 12. Ground truth changepoints for robust autoencoder (red line indicates changepoint from normal to anomalous or anomalous to normal). [Experiment 2]

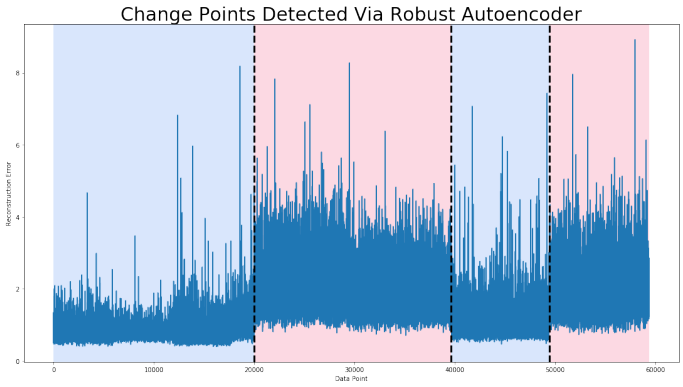


Fig. 13. Robust autoencoder correctly identifies all changepoints. [Experiment 2]

1) RPCA Performance: Figure 10 depicts the true shifts from normal to anomalous data. However, we see in figure 11 that the changepoint detection algorithm incorrectly flags some changepoints. The first changepoint (between blocks 1 and 2) is due to imbalance in reconstruction error between classes; thus, although this data (blocks 1 and 2) is normal, the algorithm flags it as "different." This issue arises again in the third changepoint (between blocks 3 and 4) due to an imbalanced representation of different classes. This experiment highlights a critical shortcoming of PCA-based methods: imbalanced representation of different classes. Possible explanations for this phenomenon include: (1) imbalanced datasets (2) not explicitly minimizing a cost function over all data.

2) Robust Autoencoder Performance: Figures 12 and 13 show that the changepoint detection algorithm on the autoencoder-based reconstruction error correctly identifies all 3 changepoints. The main difference between the autoencoder and PCA is that the autoencoder avoids false detections of changepoints, as it learns a latent representation that encapsulates the normal data more uniformly.

VI. Conclusion

Anomalous pattern detection in high dimensional spaces is a nontrivial task; many researched methods such as KNN's, Mahalanobis Distance based clustering, Local Outlier Factor, Gaussian Mixture Models and Isolation Forests, unfortunately fail to scale effectively [5]. While methods such as spectral clustering and PCA help combat the curse of dimensionality, spectral clustering may fail to detect subtle patterns in data and PCA may raise false positives due to improper lower dimensional representations. We believe robust autoencoders have 3 main benefits over current methods in the radiation safety domain: (1) they capture correlations between features unlike PCA-based techniques (2) they allow for nonlinear transformations (3) they minimize a cost function specifically designed for reconstructing points and dealing with class imbalance. Due to the large volume and high dimensional nature of radiation data, deep learning based approaches for data inference tend to be superior to traditional methods [10]. We believe the robust autoencoder and change point detection framework will help detect minor

malfunctions with higher accuracy and lower false positives.

VII. Future Work

Our initial results show promise for a more effective monitoring system of RPMs using robust autoencoders. However, we would like to perform a more comprehensive analysis of current approaches and our methodology. For example, we would like to explore different kinds of noise, compare performance to varying amounts of bias/noise, and develop a more robust metric for evaluation. Another natural step would be evaluating performance of online anomalous pattern detection.

Acknowledgments

Thank you to Professor Artur Dubrawski and Dr. Kyle Miller for their helpful guidance throughout this project. Special thanks to Rachel Burcin, John Dolan, and other organizers of the 2019 Robotics Institute Summer Scholars (RISS) program. Thank you to the Auton Lab for a memorable experience, especially Andy, Carl, and Mononito. Thank you to the National Science Foundation (NSF) and the Carnegie Mellon Robotics Institute for funding this project.

References

[1] R. Chalapathy, A. Krishna Menon, and S. Chawla, "Robust, deep and inductive anomaly detection," 04 2017.

[2] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 665–674.

[3] S. Labov, "Ernie performance with tsa portals initial results."

[4] I. Ben-Gal, "Outlier detection," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 131–146.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful?" in *International conference on database theory*. Springer, 1999, pp. 217–235.

[6] 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*, vol. 53, no. 3, pp. 1418–1423, June 2006.

[7] D. Boardman, M. Reinhard, and A. Flynn, "Principal component analysis of gamma-ray spectra for radiation portal monitors," *IEEE Transactions on Nuclear Science*, vol. 59, no. 1, pp. 154–160, Feb 2012.

[8] H. E. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1085–1089.

[9] G. C. Linderman and S. Steinerberger, "Clustering with t-sne, provably," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 2, pp. 313–332, 2019.

[10] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[11] A. P. Valentine and J. Trampert, "Data space reduction, quality assessment and searching of seismograms: autoencoder networks for waveform data," *Geophysical Journal International*, vol. 189, no. 2, pp. 1183–1202, 2012.

[12] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. ACM, 2014, p. 4.

[13] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991. [Online]. Available: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209

[14] C. Truong, L. Oudre, and N. Vayatis, "ruptures: change point detection in python," *arXiv preprint arXiv:1801.00826*, 2018.

[15] C. Rohrbeck, "Detection of changes in variance using binary segmentation and optimal partitioning," 2013.

[16] J. Caniou, "Noise in radiation detectors," in *Passive Infrared Detection: Theory and Applications*, 1999, pp. 478–520.

[17] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, "Rule-based anomaly pattern detection for detecting disease outbreaks," in *AAAI/IAAI*, 2002, pp. 217–223.

Dynamic Task Allocation Using Multi-Agent Mobile Robots

Raghavv Goel[1] Jaskaran Singh Grover[2] Sha Yi[2] Sumit Kumar[3] Katia Sycara[4]

***Abstract*— Multi-agent systems are being widely used in scenarios like search and rescue, surveillance of danger zones, monitoring of crops, pursuit-evasion etc. All these tasks/missions are dynamic, that is, the environment is constantly changing and is uncertain. Not only this, but missions may require robots of varying capability (UGVs and UAVs) to cooperate and collectively complete tasks. We thus propose an algorithm to allocate robots to tasks present in the environment as and when they discover these tasks or become aware of these tasks. Multi agent system provides wide-area coverage, fault tolerance, and robustness to missions and hence they are being preferred. While the recent work has ignored the concept of unknown environment others do not care of the path the robots will follow from one task to another. We thus integrate our method with barrier certificate, to ensure that no inter agent collision takes place and at the same time restricting the maximum inter agent distance to prevent communication loss. A team of homogeneous and heterogeneous robots has been used for the same and successful completion of all the tasks has been observed. Another concept that we introduce in this paper, is that of white agents, these are agents which can model their behaviour according to other agents depending on the task requirement(s). Lastly, we look into a learning based technique for allocating mobile tasks to the robot team, that is, these tasks are like preys which need to be captured by our agents which are predators. We do not form any comparison between model-based approach and learning-based approach as these two are very different from each other, especially in terms of the environment settings, although the inherent problem is the same.**

multi agent system, task allocation, multi-agent learning

I. Introduction

In recent times, a surge in multi-agent research has taken place. This trend has been seen in both research schools: pure controls and pure learning, especially using deep reinforcement learning [1]. Apart from the plethora of applications of multi-agent systems, some mentioned in abstract, there are limitations too. This includes having a central controller instead of a decentralized controller, which leads to high computation power and latency, poor communication due to low bandwidth and high inter-agent distances and collision avoidance. [2] introduced the concept of barrier certificate to avoid inter-agent collisions in robot teams. This concept has also been used for limiting maximum inter agent distance from another agent to prevent communication loss and thus agent loss. Having these low level controller and separate planner provides modularity.

In the task allocation problem, researchers have put emphasis on planning only, rather than control, for example [3] has done work similar to ours by using a team of heterogeneous agents, but the environment was explored beforehand, and agents were made aware of all the tasks present. They used MCST(Monte Carlo search tree) to assign the agents with tasks depending on which agent had better chances. Here, a task was defined by a set consisting of entries from one or more agents which could complete that task. The problem was solved by maximizing the individual rewards of all agents towards the total reward of the swarm. [4] proposed allocating several agents from one task to another based on a desired distribution of robots for different tasks. The robots move from one task to another based on a transition probability. In this also, the team is heterogeneous, but the agents know about all the tasks present in the environment.

We propose, to solve the problem of task allocation in an unknown environment setting with limited observability, that is only certain positions in the environment make specific tasks visible, without coming to these locations the agents remain oblivious to the fact that these tasks existed. Once a task is discovered, an optimizing problem is solved to allocate agents based on the current requirement, not every agent maybe allocated to a task, so these agents move slow till it is confirmed that no more agents will be required. More agents can be required as there exists a possibility that the agents assigned to a task discover another task visible via the current task they have gone to but not visible to the others. This information is communicated back to other agents, as communication exists between agents as they always remain within the communication distance of at least one other agent.

[5] have formed an optimization problem for different types of agents, for finding the best step size for each type of agent. In all of these works, all the agents are just transported from one task to another, ignoring the path traversal and the challenges it brings. As the swarm size increases the path traversal becomes difficult because of less space for an agent and avoiding collisions can influence the planner. Here, merely forming solutions for assigning agents from one task to another does not solve the challenges faced in the real world, where the robot needs to travel to the new location of the new task physically.

In multi-agent-based learning, a lot of work has been done, such as [1], [6]. We examine and change [6], which was modeled only for static tasks, to mobile tasks. MAPE (multi-agent particle environment) [7] has been used for simulating

[1] was a Robotics Institute Summer Scholar and is a student at IIIT Delhi, India, raghav16179@iiitd.ac.in

[2] are PhD students at Robotics Institute, Carnegie Mellon University, jaskarag@andrew.cmu.edu, shayi@andrew.cmu.edu

[3] is a masters student at Robotics Institute, Carnegie Mellon University, sumitsk@cmu.edu

[4] is a research professor at Robotics Institute, Carnegie Mellon University katia@cs.cmu.edu

the algorithm and even designing new environments for enabling mobile task allocation, which is described later in the methods section. The agents and the targets(tasks) are considered as a connected graph, with all agents connected to all the tasks at all times but with varying inter-agent connectivity. The graph structure was used because of the inherent property of a graph to ensure permutation invariance. The observation of all the neighbors of an agent is aggregated and sent without any order. Permutation invariance is important to prevent bias due to agent order. Furthermore, all the observation of the neighboring agents is passed via the attention model [8] so that agents can have preferences for neighboring observations.

II. Model Based

We have a mission which needs to be completed, here the mission is to check all the rooms as shown which has a number of rooms and walls(black), with a team of robot agents also present. This team of robots needs to go into all the rooms and check these rooms. Checking a room can be related to completing a task present in a room like attacking enemies or helping a human in need or just scouting the area. We use a team of robots with mixed capabilities, like for example ground vehicle(Clear-path Husky), aerial vehicle(Hexacopter) and underwater vehicle, which we denote using different colors circles: Red, Green, and Blue. The catch of having different color rooms is that it can be checked only by those color robots. In the real world, often mission requires cooperation between UGVs and UAVs to increase the overall efficiency of the mission [9].

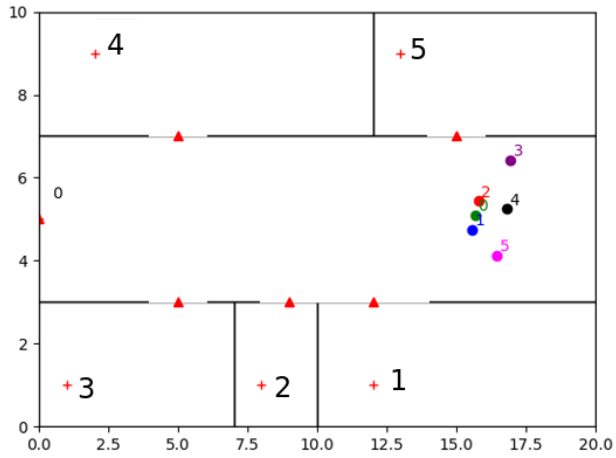


Fig. 1: Environment without leaf room and same rooms, used for testing our proposed method. The red triangles are mid points of doorways while the red plus(+) signs are goal positions of rooms.

III. Environment

We have fours different scenarios as shown in table I. The basic setting as shown in figure 1 consists of homogeneous agents and tasks of a varying number of agent requirement without any leaf room (defined below), while the most difficult setting has heterogeneous agents with tasks of different capabilities, varying number of agent requirements and having leaf rooms as shown in figure 2. A leaf room is a room which is not accessible from the corridor which is the root room. Table I shows the different settings which will be covered.

TABLE I: Different scenarios

	Type of Rooms	Type of Agents	Req. agents per room	Leaf Rooms
1.	same color	same color	fixed	NA
2.	diff color	RGB	fixed	NA
3.	diff color	RGB	varying	A
4.	diff color	RGB + white	varying	A

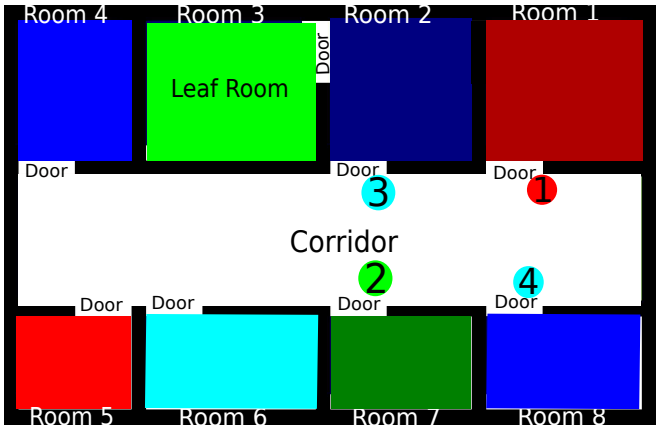


Fig. 2: These rooms are tasks present in the environment (which need to be completed), the room color denotes the type of agent it needs and the darker a room is the more number of agents is required.

IV. Preliminaries

The method for the basic scenario 1, refer to figure 1, is mentioned first, and then the rest of the scenarios are built on top of it. Below are definitions which are used in the algorithm 1 proposed by us.

We assume the origin to be the top left corner of figures shown.

1) M = total number of room excluding the root room
2) N = total number of robots/agents
3) $\hat{e}_i \in \mathbb{B}^{(M+1)\times 1}$ with i^{th} entry as 1, and rest as 0.
4) $\hat{\mathbf{e}}_i \in \mathbb{B}^{N\times 1}$ with i^{th} entry as 1, and rest as 0.
5) $R = \{0, 1, 2, \ldots, M\}$ where each number denotes the ID of the room assuming there are M rooms.
6) $R = \begin{bmatrix} R_0 & R_1 & \ldots & R_M \end{bmatrix}$ where $R_i \in [(x^{R_i}_{leftmost}, y^{R_i}_{topmost}), (x^{R_i}_{rightmost}, y^{R_i}_{bottommost})]$ $\forall$ i $\in \{1, 2, \ldots, M\}$, R_0 is the corridor which is connected to all the rooms and hence no specific door is required.
7) $doors = \begin{bmatrix} d_{01} & d_{02} & \ldots & d_{0M} \end{bmatrix}$ where d_{ij} is a door between R_i and R_j, $i < j$. $d_{ij} \in [(x^{ij}_{left}, y^{ij}), (x^{ij}_{right}, y^{ij})]$ assuming that doors will be parallel to x axis.
8) $G = \begin{bmatrix} g_0 & g_1 & g_2 & \ldots & g_M \end{bmatrix}$, where $g_i \in \mathbb{R}^{2\times 1}$ is

the goal position of $R_i \; \forall$ i $\in \{0, 1, 2, \ldots, M\}$, which implies that $G \in \mathbb{R}^{2\times(M+1)}$

9) $P = \begin{bmatrix} p_0 & p_1 & p_2 & \ldots & p_M \end{bmatrix}$, where $P_i \in \mathbb{R}^2$ is the corresponding point in corridor(R_0) which the robot(s) returns to after checking R_i at G_i. As g_0 is already present in the corridor, hence $p_0 = g_0$
10) $Q^t = \begin{bmatrix} q_1 & q_2 & \ldots & q_N \end{bmatrix}$, where $q_i \in \mathbb{R}^{2\times1}$ is the corresponding position of the i^{th} robot at time $t \; \forall \; i \in \{1, 2, \ldots, N\}$, which implies that $Q \in \mathbb{R}^{2\times N}$
11) $A = \{1, 2, \ldots, N\}$ is the set of all robots, assuming there are N of them.
12) $R_A \in \mathbb{B}^{(M+1)\times1}$, where each element is $\in \{0, 1\}$, depending on the following

$$R_A(j) = \begin{cases} 1 & \text{if } \exists \; i \in \{1, 2, \ldots N\} \text{ s.t. } e_x Q^t \hat{\mathbf{e}}_i \in \\ & [x_{R_j - door - left}, x_{R_j - door - right}] \text{ and} \\ & e_y Q^t \hat{\mathbf{e}}_i \in [y_{\text{corridor-bottom}}, y_{\text{corridor-top}}] \\ 0 & \text{otherwise} \end{cases}$$

Where, $j \in \{1, 2, \ldots M\}, e_x = [0 \;\; 1]$ and $e_y = [1 \;\; 0]$ are $\in \mathbb{R}^{1\times2}$. This is updated after every time step whenever an agent enters the doorway of a room not already present.
Once a room is discovered by an agent, i.e., it becomes available, then at no point of time, it becomes undiscovered.
$R_A(0) = 1$ from the start, as the agents where the corridor ends. Please refer to figure 2 which shows no walls present between the agents start and the end of the corridor, so it is assumed that the agents know where the goal of R_0 is.
It can be accounted in future works that the agents are given some sort of memory in which an available room, if left unchecked for a long time may go unavailable as well.
13) $R_C \in \mathbb{B}^{M\times1}$, where each element is $\in \{0, 1\}$, depending on the following.

$$R_C(j) = \begin{cases} 1 & \text{if } \exists \; \{i_1, i_2, \ldots\} \in \{1, 2, \ldots N\} \text{ s.t.} \\ & Q\hat{\mathbf{e}}_{i_l} = G\hat{e}_j, \forall l \in \{1, 2, \ldots |R_{req}(j)|\} \\ 0 & \text{otherwise} \end{cases}$$

This is updated whenever a robot clears a room.
14) $R_{req} \in \mathbb{N}^{M\times1}$: which tells the requirement of M rooms and the corridor(0^{th} room) where R^i_{req} is the requirement of the i^{th} room where $i \in R$.
15) K is a scalar and is the total number of rooms available but not checked, it computed as follows:

```
K = 0
for j <- 1 to M do
    if RA(j) == 1 and RC{j} == 0 then
        K += 1
    end if
end for
```

16) $T \in \mathbb{B}^K \times \mathbb{B}^{M+1}$, is a transformation matrix which gives all the available and unchecked rooms when multiplied with other matrices mentioned below.
Here $k = 0$ initially and then $k = k+1$, while $k < K$

$$T(k,:) = \begin{cases} \hat{e}_j^\intercal & \text{if } R_A(j) = 1, R_C(j) = 0 \\ 0 & \text{otherwise} \end{cases}$$

17) $X^t \in \mathbb{B}^{M+1} \times \mathbb{B}^N$, is the assignment matrix:

$$X^t(j,i) = \begin{cases} 1 & \text{if } Q^t\hat{\mathbf{e}}_i = G\hat{e}_j \\ 0 & \text{otherwise} \end{cases}$$

Where, $j \in \{0, 1, 2, \ldots, M\}$and $i \in \{1, 2, \ldots, N\}$which tells which robot is in which room.
18) $U^t \in \mathbb{R}^{(M+1)\times1}$, a utility vector which helps agent decide whether to enter a room or not. It is defined as:

$$U^t(j) = \begin{cases} 0 & \text{if } j = 0 \\ 2 & \text{if } R_A(j) = 1, R_C(j) = 0, \\ -2 & \text{if } R_A(j) = 1, R_C(j) = 1, \\ \text{Don't Care} & \text{if } R_A(j) = 0, R_C(j) = 1, \\ \text{Don't Care} & \text{if } R_A(j) = 0, R_C(j) = 0, \end{cases}$$

The values assigned above are for specific reasons so that the agents move in the desired way. The corridor is like a neutral ground for all the agents and hence given the value 0. As agents should cover all the available and unchecked rooms, these rooms have been given a value of +2, motivating the agents to move in these rooms. After the agents move and check these rooms, there is no longer a need for the agents to stay in them, and so the value given to available but checked rooms is -2. This makes the value of the corridor, which is 0, greater than the checked rooms, making the agents move back to the corridor.
If a room is not available, it implies that the agents are not aware of this room's existence, so no utility can be assigned to these rooms and hence don't care has been written.
19) $D^t \in \mathbb{R}^M \times \mathbb{R}^N$:

$$D^t(j,i) = \begin{cases} -\|G\hat{e}_j - P\hat{e}_j\|_2 - \|P\hat{e}_j - PX^t\hat{\mathbf{e}}_i\|_2 - \\ \|PX^t\hat{\mathbf{e}}_i - Q^t\hat{\mathbf{e}}_i\|_2 \quad \text{if } Q^t\hat{\mathbf{e}}_i \notin R_0, \text{ else} \\ -\|G\hat{e}_j - P\hat{e}_j\|_2 - \|P\hat{e}_j - Q^t\hat{\mathbf{e}}_i\|_2 \end{cases}$$

where $j \in \{0, 1, 2, \ldots, \text{M}\}$ and $i \in \{1, 2, \ldots, N\}$
20) $dq^t \in \mathbb{R}^{2\times N}$ step size taken by an agent in the desired direction, is defined as:

$$dq^t(i) = \begin{cases} PX^t\hat{\mathbf{e}}_i - Q^t\hat{\mathbf{e}}_i & \text{if } Q^t\mathbf{e}_i \in R_0 \\ G\hat{e}_j - Q^t\hat{\mathbf{e}}_i & \text{if } Q^t\mathbf{e}_i \in R_j, GX^t\hat{\mathbf{e}}_i \in R_j \\ P\hat{e}_j - Q^t\mathbf{e}_i & \text{if } Q^t\mathbf{e}_i \in R_j, GX^t\hat{\mathbf{e}}_i \notin R_j \end{cases}$$

Where, $i \in \{1, 2, \ldots, N\}$ and
$\tilde{dq}^t = \frac{dq^t}{\|dq^t\|_2} v_{max}$ if $\|dq^t\|_2 > v_{max}$

TABLE II: SRSA matrix assignment for figure 2

	a_1	a_2	a_3	a_4
R_0	1	1	1	1
R_1	1	0	0	0
R_2	0	0	1	1
R_3	0	1	0	0
R_4	0	0	1	1
R_5	1	0	0	0
R_6	0	0	1	1
R_7	0	1	0	0
R_8	0	0	1	1

V. Methods

1 **For same room same color**
Please note that: U, D is same U^t, D^t

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \lambda_1 u^\intercal x + \lambda_2 d^\intercal x \\ \text{subject to} \quad & E_1 x = \mathbf{1}_N, \\ & TE_2 x \geq TR_{req}, \\ & E_3 x = \mathbf{0}_{M+1}, \\ & x \in \mathbb{B}^{N(M+1)\times 1} \end{aligned} \tag{1}$$

where, $\lambda_1 \in [0,1]$ and $\lambda_2 \in [0,1]$, $d = vec(D^t)$, $x = vec(X^{t+1})$, $\mathbf{u} = \begin{bmatrix} U_{\times N} \end{bmatrix} \in \mathbb{R}^{(M+1)\times N}, u = [\dot{\mathbf{e}}_j^\intercal \mathbf{u}_{\forall j \in \{1,2,\ldots,N\}}]$
$E_1 = [I_{N_{\times(M+1)}}]$
E_2 and E_3 are calculated as follows:

```
E2 = zeros(M+1, N(M+1))
for j <- 0 to (M+1) do
  E2(j, j*N: (j+1)*N) = ones(N)
end for

E3 = zeros(M+1, N(M+1))
for j <- 0 to (M+1) do
  if RA{j} == 0 then
    E3(j, j*N: (j+1)*N) = ones(N)
end for
```

The aim is to assign $\geq$ number of agents to all the available, unchecked rooms. For this an objective function is made, as shown in equation 1, the solution to this assigns agents to all the available rooms. The assignment is done by maximizing the cost function. The constraints help us to restrict the agent assignment to only one room and also take into account only the rooms which are available and unchecked to have assignments.

2 **Different color rooms, different color agents**
Here, if the color of the agent and the room is the same, then, the agent should go into the room, otherwise not. To enforce this, a matrix is defined as follows:
This is called S and remains fixed for an environment setting refer to table II. Another constraint is added to the already existing constraints for preventing different color agent(e.g., green) from entering a red room.
We introduce an adjacency matrix $\mathbf{N} \in \mathbb{R}^{(M+1)\times(M+1)}$ here, which keeps an account of which room is neighbour of which. We use this to enforce another constraint that an agent can only enter a leaf room via the room connected to it.
$\mathbf{N} \in \mathbb{B}^{(M+1)\times(M+1)}$ defined as

$$\mathbf{N}(j,i) = \begin{cases} 1 & \text{if door present} \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 1 Assigning Rooms to Robots

Initialize:
$\text{pos}_0 \longleftarrow$ Initial Position of Robots
$R_{req} \longleftarrow$ Robots requirement for each room
$\texttt{available-rooms}_{curr} \longleftarrow$ Rooms available currently

$\texttt{cleared-rooms}_{curr} \longleftarrow$ Rooms cleared currently
$\texttt{RRA}_{curr} \longleftarrow$ Current Rooms-Robot-Assignment.

while `all-rooms-not-cleared` **do**
  Calculate
    $\texttt{room-utility}_{curr} \leftarrow \texttt{available-rooms}_{curr}$
    $\texttt{room-utility}_{curr} \leftarrow \texttt{cleared-rooms}_{curr}$
  Determine $\texttt{RRA}^*_{new}$ by maximizing

Total Utility from Assignment

  subject to

room min-robot-requirement-constraint

room-capability-constraints

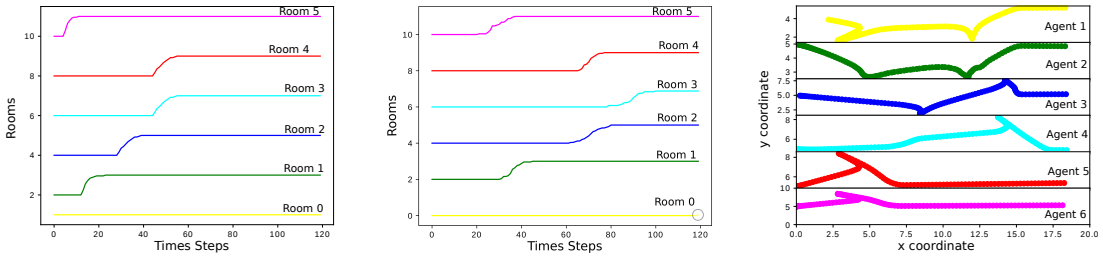
  Calculate
    Control u^* to the centroid in the newly assigned room, calculated using $\texttt{RRA}^*_{new}$ (velocity to centroid of room)
  Update
    $\text{pos}_{new} \leftarrow f(\text{pos}_{curr}, u^*)$
    $\texttt{available-rooms}_{new}$ from pos_{new}
    $\texttt{cleared-rooms}_{new}$ from pos_{new}
end while

4 **Different color rooms, different color agent, leaf rooms & white agents**
Agents with diverse capabilities are introduced; these agents are white in color and then change their color according to the environmental requirements.

VI. Learning Based

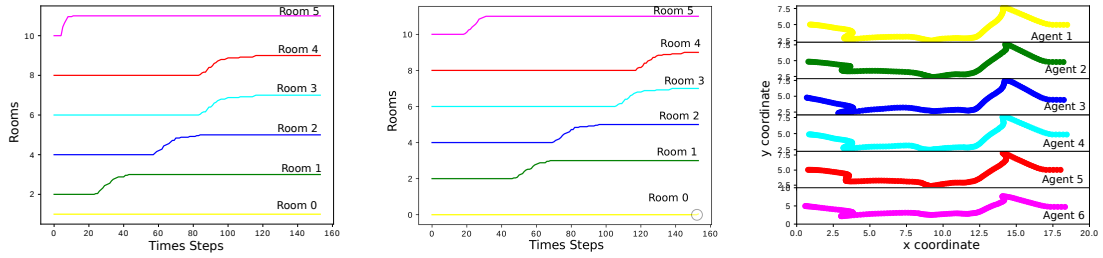
The environment is shown in figure 7 and 8, where the aim is to capture the red colored circles(preys). The dimensions of the environment is 2×2 .The movement of prey(s) is modeled like that of a snooker ball, whenever the prey hits the edge its velocity perpendicular to that edge becomes opposite. The agents(blue circles) have a double integrator control and can move in x and y with maximum acceleration = 5 and time step $\mathrm{d}t = 0.1$.

The setup with a single prey and multiple predators has the reward as the negative of the sum of distances of agents from the prey, and a trial is said to be successful if all the agents are within a threshcurrent distance (usually the sum of radii of agent and prey) of the prey.



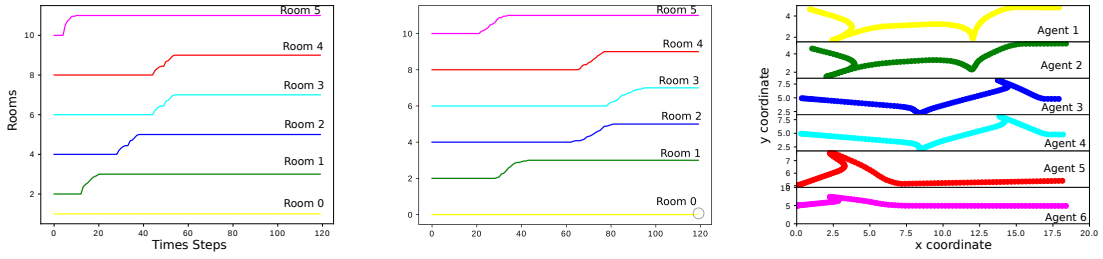
(a) (b) (c)

Fig. 3: (a) shows the time when each room becomes available, (b) shows the time when a room is checked and (c) shows the trajectories the agents follow. These follow similar pattern below and hence we will not be repeating this information below.)$\lambda_1 = 1.0, \lambda_2 = 0.0$. The average time taken to check all the rooms = 135.24 and the maximum time was = 169.



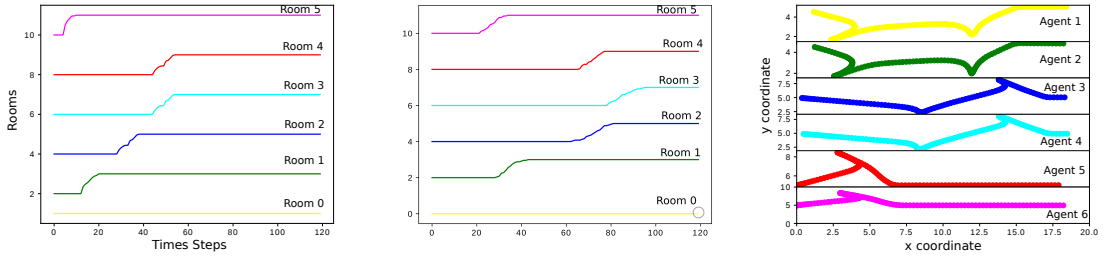
(a) (b) (c)

Fig. 4: $\lambda_1 = 1.0, \lambda_2 = 0.0$. The average time taken to check all the rooms = 162.92 and the maximum time was = 182.



(a) (b) (c)

Fig. 5: $\lambda_1 = 1.0, \lambda_2 = 0.0$. The average time taken to check all the rooms = 126.28 and the maximum time was = 136.



(a) (b) (c)

Fig. 6: $\lambda_1 = 1.0, \lambda_2 = 0.0$. The average time taken to check all the rooms = 127.04 and the maximum time was = 136.

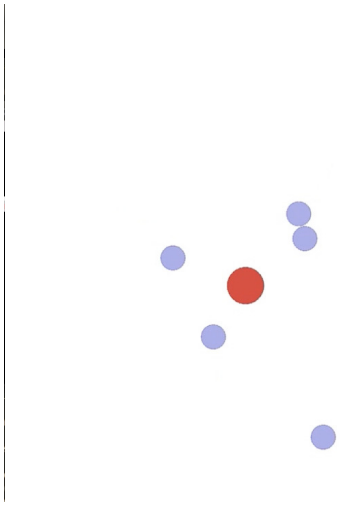


Fig. 7: The blue circles are agents(predators) and the red circle are tasks(preys), which need to be captured

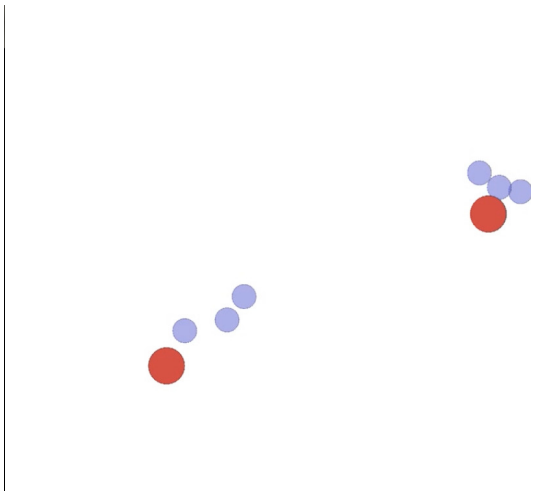


Fig. 8: Similar to figure 7, but here the agents need to collaborate to distribute the task of capturing the preys.

For the multiple preys case, along with the above, additional positive reward needed to be given for making the agents split into small teams to capture multiple preys at the same time instead of all agents running behind a single prey. As preys require a certain number of agents to get captured, the done/success condition was also modified accordingly.

Testing speed	Success (out of 100)	Avg. episode length (out of 50)
0.025	100	19.30
0.05	70	35.02
0.075	0	50

TABLE III: 3 agents and 1 task, trained on a constant velocity of 0.025 in both x and y but tested on different velocities

Testing speed	Avg. ep. (max 50)	Success (max 100)	Avg. ep. (max 100)	Success (max 100)
0.025	61	35.96	75	52.56
0.050	92	25.57	100	26.89
0.075	21	46.81	46	80.86
0.100	0	50	0	100

TABLE IV: 3 agents and 1 task, trained on a constant velocity of 0.05 in both x and y but tested on different velocities

Collision at training	Collision at testing	Avg. Ep. (max 100)	Sucess (max 100)	Min distance req. for capture
Yes	Yes	100.00	0	0.125
No	Yes	100.00	0	0.125
Yes	No	96.58	12	0.125
No	No	93.59	20	0.125
No	No	83.44	44	0.15

TABLE V: 6 agents and 2 tasks, trained on a constant velocity of 0.025 in both x and y and tested on same velocity but we toggle the inter agent collision and agent to task collision

VII. Conclusion

For the model based case, discussed before the learning based control, we ran our proposed algorithm for scenario 1 and scenario 2 as mentioned in table I. We tested for two cases, $\lambda_1 = 1, \lambda_2 = 0$ and $\lambda_1 = 0, \lambda_2 = 1$ and observe for scenario 1 that giving higher preference to utility leads to faster coverage of all the rooms as compared to giving preference to distance matrix. (a) shows the time when each room becomes available, (b) shows the time when a room is checked and (c) shows the trajectories the agents follow. All of them have been averaged over 25 runs. For rooms available and checked, we show that if a room is available then it's value becomes one in its corresponding row, as shown in figures 3, 4, 5, 6. For the scenario 2, refer to 5, 6, we see a very similar trend between both the cases. One for reason for this can be that due to limited number of rooms, the restriction on agent color and room color leads to exploration even in the case where $\lambda_2 = 1, \lambda_1 = 0$, which has only distance exploitation. If given a large number of rooms in parallel, then it'll be interesting to see the result of both the cases. We observe that higher dependence on utility u i.e. λ_1 closer to 1, leads to faster coverage versus distance. However, when we introduce room color constraint then both the cases have almost equal maximum time taken to check all the rooms and average checking time is same. We still need to integrate our latest approach with barrier certificate to prevent inter agent collisions. Also, we need to further build for last two scenarios. Bringing this method to real world can become costly as the number of robots increases, as then solving the optimization will take more time, so we need to find a way to solve the optimization after certain interval of times and still giving the same result.

VIII. Future Work

An interesting setting in the model-based case will have a series of leaf rooms, one after the other. Here, agents having

more capabilities similar to the white agents mentioned earlier will be immensely useful as they can adapt according to the room requirements versus, the normal agents which move slowly when they are passing the doorways. Because, as the doorways within a room increases, the time delay due to the slow-motion does not give the agent enough time to explore every room. Furthermore, testing on more complicated maps like those having loops will also be interesting.

In the learning-based approach, one thing that will help improve the accuracy of getting close to the tasks as soon as possible is to introduce a recurrence model which can store a series of positions of the tasks, and the agents can then approximate a function to predict the next position of the task better. Also, making the environment more complicated by introducing obstacles and having a heterogeneous team are also interesting avenues. In terms of having a heterogeneous team, the agents can have different underlying controller: single integrator versus a double integrator, the map can have areas which can be restricted for one type of agent and not for the other and the velocities limits are some of the traits which can played with and is even useful in the perspective of real-world, where for example: a building has several entries, but entries high above the ground cannot be easily accessed by aerial vehicles instead of ground vehicles, similarly collapsed building often have small entries near the ground where ground vehicle can work really well.

ACKNOWLEDGMENT

I would love to thank the AART Lab for having me as a summer intern and giving me this awesome opportunity. A big thanks to Dr. Sycara and all the lab members who supported me throughout the summer in some way or the other, some of whom I couldn't mention in the authors are mentioned here: Keitaro, Akshay, Swaminathan, Dana, Michael, Simran, Sidharth, and Azan.

REFERENCES

[1] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[2] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[3] A. J. Smith, G. Best, J. Yu, and G. A. Hollinger, "Real-time distributed non-myopic task selection for heterogeneous robotic teams," *Autonomous Robots*, vol. 43, no. 3, pp. 789–811, 2019.

[4] A. Prorok, M. A. Hsieh, and V. Kumar, "Fast redistribution of a swarm of heterogeneous robots," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and , 2016, pp. 249–255.

[5] L. Mo, J. Li, and J. Huang, "Distributed optimization algorithm for discrete-time heterogeneous multi-agent systems with nonuniform stepsizes," *IEEE Access*, vol. 7, pp. 87 303–87 312, 2019.

[6] A. Agarwal, S. Kumar, and K. Sycara, "Learning transferable cooperative behavior in multi-agent teams," *arXiv preprint arXiv:1906.01202*, 2019.

[7] https://github.com/openai/gym.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[9] P. Maini, S. PB, and P. Tokekar, "Visual monitoring for multiple points of interest on a 2.5 d terrain using a uav with limited field-of-view constraint," *arXiv preprint arXiv:1903.07363*, 2019.

Discriminating Cognitive Disequilibrium and Flow in Problem Solving: A Semi-supervised Approach Using Involuntary Dynamic Behavioral Signals

Mononito Goswami[1], Lujie Chen[2] and Artur Dubrawski[2]

***Abstract*— Problem solving is one of the most important 21st century skills. However, effectively coaching young students in problem solving is challenging because teachers must continuously monitor their cognitive and affective states, and make real-time pedagogical interventions to maximize their learning outcomes. It is an even more challenging task in social environments with limited human coaching resources. To lessen the cognitive load on a teacher and enable affect-sensitive intelligent tutoring, many researchers have investigated automated cognitive and affective detection methods. However, most of the studies use culturally-sensitive indices of affect that are prone to social editing such as facial expressions, and only few studies have explored involuntary dynamic behavioral signals such as gross body movements. In addition, most current methods rely on expensive labelled data from trained annotators for supervised learning. In this paper, we explore a semi-supervised learning framework that can learn low-dimensional representations of involuntary dynamic behavioral signals (mainly gross-body movements) from a modest number of short time series segments. Experiments on a real-world dataset reveal a significant advantage of these representations in discriminating cognitive disequilibrium and flow, as compared to traditional complexity measures from dynamical systems literature, and demonstrate their potential in transferring learned models to previously unseen subjects.**

I. Introduction

One of the fundamental goals of education is to transform students into mature problem solvers who are able to overcome the inherent uncertainty of problems, failed attempts and impasses. For young children, solving challenging non-routine math problems emulates the real life challenges they will encounter later in their lives. Different from routine math exercises (e.g. back-of-chapter exercises), non-routine problems may not have immediate solutions, and thus require innovative thinking, and may often invite a child to ride an "emotional roller-coaster" as the student advances through various stages of problem solving [1]. Problem solving is a complex affective and cognitive process replete with states of *cognitive disequilibrium* manifested by a mixture of confusion, frustration, indecisiveness or struggle, as well as states of *flow* [2] when one is (or at least is feeling of) moving forward smoothly. The cognitive disequilibrium triggered by conflicts and contradictions in these problem solving processes can be beneficial for learning only if appropriately regulated and resolved [3] (*Facilitative Confusion Hypothesis*), which may be challenging for an inexperienced problem solver whose self-regulation and problem solving skills are in their nascent stages.

Therefore, effectively coaching young students requires teachers to continuously monitor their cognitive and affective states and make real time pedagogical decisions such as when to intervene and how best to do so, especially in social environments with low teacher-student ratios and with limited coaching resources available for each student. Moreover, teachers also have to effectively handle the high cognitive loads of monitoring a diverse cohort students varying significantly in their perception of academic self-efficacy and ability to use of self-regulated learning strategies [4]. Intelligent Tutoring Systems that attempt to teach problem-solving also face similar challenges. To lessen the cognitive load of teachers and also to improve the effectiveness of intelligent tutoring, we envision a decision support system which can monitor the cognitive and affective states of multiple students simultaneously in real time. The focus of this paper is on the state detection capability of such a system, specifically needed to discriminate between cognitive disequilibrium (CD) and flow states, which are the critical inputs to inform appropriate subsequent interventions.

In this work, we investigate a method designed to discriminate between CD and flow using involuntary behavioral signals that are less prone to social editing, including head and eye movement, which can be non-invasively collected using inexpensive sensors such as cameras. To overcome limited supply of labeled data, while taking advantage of the large supply of unlabeled data, we explore a semi-supervised approach where deep embedding features are derived from unlabeled time series segments, which are then fed into a supervised learning algorithm. We compare these deep features with a set of baseline complexity measures discussed in dynamical systems literature and note significant improvement in predictive power. Furthermore, our experiments confirm that our semi-supervised model performs reasonably well even with very limited amount of data.

The rest of the paper is organized as follows. Section II provides background of our study by discussing its motivation and relation to prior work. Section III describes the data collection, methodology, and experiments in detail. Section IV discusses experimental results, and Section V explores their implications. We conclude the paper and

[1]M. Goswami is a senior student in Computer Science & Engineering Department, Delhi Technological University, New Delhi, India mononitog@hotmail.com

[2]Lujie Chen & Artur Dubrawski are with the Auton Lab, Robotics Institute, Carnegie Mellon University, USA karenchen, awd@cs.cmu.edu

present avenues of future work in Section VI.

II. Background and Related Work

A. Cognitive Disequilibrium and Flow in Problem Solving

In the last few years, the research community has shown keen interest in the affective and cognitive dimensions of learning [5]. Studies such as [3] have shown that children may get *confused* when they are unsure of how to proceed or face challenging impasses. They may also get *frustrated* when they repeatedly make mistakes or important goals are blocked [6]. At the same time, students may also experience *delight* when they achieve their goals by overcoming problems or enter a *flow* state of intense engagement when the learning goals as well as problem solving paths are clear, and they find an appropriate balance between skills and challenge [2].

In the practice of problem solving education, a teacher would be broadly interested in two cognitive states of the student: (1) *Cognitive Disequilibrium*, characterized by confusion, frustration and indecisiveness, and (2) the *Flow* state, characterized by smooth progression toward the goals. In this paper, we only consider two distinct and broad cognitive states of primary interest to teachers, and therefore we also attribute positive emotions such as curiosity and happiness to the flow state, and pose the problem of detecting cognitive states of a student as a binary classification problem. [5] previously found that flow, confusion and boredom were the most frequent affective states found in studies employing learning with technology, in support of our choice of the cognitive-affective states to consider.

B. Automated Detection of Cognitive-Affective States

The problem of identifying cognitive and affective states of students is challenging since these states are loosely-defined psychological constructs embedded in extremely context-sensitive environments [7]. Many studies have investigated the possibility of detecting affective states automatically, primarily in the context of Intelligent Tutoring Systems. For instance, [8] proposed *Engagement Tracing* to detect the engagement levels of students based on audit logs from an intelligent tutor. Later, [9] investigated the relationship between facial features and emotions such as confusion, frustration and delight, and found important patterns in the way that learners communicate their emotions through their faces. Most recent literature on affective and cognitive computing has focused on the use of multimodal features. These multimodal affect classifiers have also been shown to be consistently better than their unimodal counterparts [10]. [11] in their affect-sensitive AutoTutor combined decisions from conversational cues, gross body language and facial feature tracking in order to track the affective and cognitive states of students. [12] used multi-channel physiological signals such as heart-activity, skin conductivity and respiration to detect the learner's affective states during their interaction with AutoTutor. While many of these classifiers have achieved impressive performance, one major limitation is their reliance on data from *expensive* and *intrusive* sensors to monitor body posture (Body Posture Management System), electrocardiogram (ECG), etc. *The expense of these sensors coupled with their intrusive nature preclude their deployment at scale, in common classrooms, and in less developed communities.* Furthermore, many studies have used facial expressions and vocal features as indicators of affect [13]. While facial expressions are widely considered as a language of emotion [14], many studies such as [15] have highlighted that culture and ethnicity may influence the recognition of emotion by facial expressions. Furthermore, most affective classifiers are trained using supervised machine learning and require a sufficient supply of labeled *"ground-truth"* data from experts and self-reports. Obtaining labeled data free from cultural, reference [16] and social desirability [17] is very hard. In such a scenario, unsupervised representation learning methods may be handy, since they do not require training data and may learn useful features. We illustrate the feasibility of semi-supervised models in the cognitive state detection pipeline through our experiments later in the paper.

C. The Expressive Power of Gross Body Movements

Many researchers have investigated the role of facial expressions, speech patterns and physiological responses, as indices of cognitive and affective states. [18] also pointed out that owing to the numerous degrees of freedom, the human body is a potentially ideal affective communication channel. However, only few studies have focused on *gross-body movements* as predictors of cognitive and affective states, which is surprising due to the embodied nature of affect and cognition [18]. Gross-body movements are promising predictors of cognitive states because they are mostly involuntary and therefore less prone to social editing in comparison to vocal and facial features. In addition, the human body owing to its large number of degrees of freedom forms a rich affective communication channel [18]. Existing research utilizing gross body movements as an index of affect, has mostly focused on gestures and specific postures [19], and relied on expensive sensors such as Body Posture Measurement Systems [20], which are hard to deploy at large scales in practice. A few years ago, [18] established that body fluctuations in the normal state of mind (cognitive equilibrium) are characterized by *correlated pink noise*, and underwent *whitening* when their participants experienced states of cognitive disequilibrium. Inspired by the findings, we hypothesized that states of cognitive disequilibrium and flow differ in the complexity of the gross body movement signals. By considering gross body movements in addition to facial action units in the form of time series (rather than raw video logs), we ensure that our features are not only privacy-preserving, but also involuntary and therefore less susceptible to social editing.

III. Data and Methodology

A. Data Collection and Pre-processing

Our experiments are based on a dataset collected in one-to-one coaching scenarios for math problem solving. Seven children within eight to twelve years of age and their parents

were recruited from a local community. Parents were asked to record videos (using a web camera) and pencast videos (using a Livescribe Smartpen) of their children solving a math problem. The cohort comprised of three girls, four boys (two girls were siblings) and their parents (two fathers and four mothers). The dataset consisted of 36 sessions having a cumulative duration of 307 minutes, with a mean duration of 7.9 minutes per session.

A number of features were extracted from the dataset along the visual and writing channels. Visual features such as Facial Action Units (FAUs), head and eye gaze orientations were extracted using OpenFace [21] at a sampling frequency of 30 Hz. We computed the first and second order derivatives of all visual features with the exception of Facial Action Units using NumPy's gradient function which approximates the gradient of an array using second order accurate central differences in the interior points and second-order accurate one sides in the end points. The writing speed was estimated from Livescribe Echo Smartpen by computing the cumulative distance covered by the tip of the pen and thereafter measuring the change in the "amount of ink" collected in a trailing window of two seconds. The final sets of features used in our study are listed in Table I.

B. Ground Truth Labels

In order to validate our results, we annotated non-overlapping 10-second time series segments for states of cognitive disequilibrium or flow. 20% of the video segments from each child were annotated by two independent annotators[1]. Each annotator labeled a ten-second window within a session based on the "perceived" cognitive state of the child, as *cognitive disequilibrium* (1), *neutral* (2), *flow* (3) or *off-task behavior* (-1). The rest of the data were then labeled by one annotator, after a satisfactory inter-rater consensus was reached with Cohen's kappa greater than 0.5.

The choice of 10-second windows was inspired by literature where a number of studies such as [18] used fixed size windows for annotating affect. The choice of the window size was also driven by the fact that complexity measures such as Higuchi Fractal Dimension expect stationary time series as input, and while 10s windows (300 time steps at 30 frames-per-second video) are short enough to be considered stationary, they also include sufficient number of time steps to accurately compute the complexity measures. In our experiments, we only used time series segments labeled as cognitive disequilibrium and flow since both the annotators had substantial agreement (average Cohen's kappa = 0.6). From a total of 353[2] time series segments, we could only use 248[3] time series segments for our analysis. The remaining segments were shorter than 10s, too short for computing complexity measures.

[1]The first and second authors of the paper. The second author has considerable experience in annotating similar datasets.

[2]198 segments for Cognitive disequilibrium and 158 for flow

[3]128 segments for Cognitive disequilibrium and 120 for flow

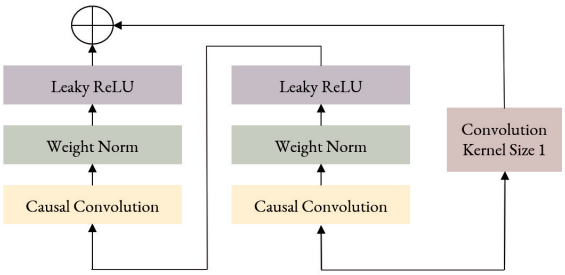


Fig. 1. Composition of the i-th layer of the network [30].

C. Measures of Time Series Complexity

Measures of time series complexity were developed to distinguish regular, chaotic and random behavior. Measures such as Higuchi Fractal Dimension, Approximate Entropy, etc. have been widely used in bio-medical signal processing applications such as electroencephalographic time series analysis [22] and psychology [23].

In their seminal work, [18] found that fluctuations in gross body movements in states of cognitive equilibrium are characterized by *correlated pink noise*, and undergo *whitening* when students experience cognitive disequilibrium. White noise is characteristic of random systems having no long or short term correlations between observations, whereas pink noise exhibits both long and short term correlations [18]. Inspired by those results and the success of complexity measures in analyzing physiological time series [24], we hypothesize that states of cognitive disequilibrium and flow may differ in complexity.

Numerous time series complexity measures have been proposed, but in our study we consider the following six most widely used: *Approximate Entropy* [25], *Sample Entropy* [26], *Spectral Entropy* [27], *Permutation Entropy* [28], *Katz Fractal Dimension* [22], and *Higuchi Fractal Dimension* [29].

In order to test our hypothesis, we conducted the two-sample Kolmogorov-Smirnov (K-S) test which compares the empirical distribution functions of two samples under the null hypothesis that both are drawn from the same underlying distribution. We carried out a total of 48 univariate two-sample K-S tests, one for each combination of 8 features (*gaze_vel_X, gaze_vel_Y, gaze_acc_X, gaze_acc_Y, head_vel_T, head_vel_R, head_acc_T, head_acc_R*) and 6 complexity measures. The two samples for the test were the states of cognitive disequilibrium and flow. We also carried out a randomization test (with 1000 runs) and computed the K-S statistics (D) by randomly permuting cognitive state labels. The results of our experiments are discussed in detail in Section IV-A.

D. Deep Feature Embedding

The field of affective and cognitive computing relies on supervised learning algorithms [31], and is therefore heavily dependent on training data from expert annotators or self-reports by participants of a study. Since most advanced

Features	Description	Derivation from OpenFace features
FAUs	Indicate the presence or absence of 18 Facial Action Units	AU01_c, AU02_c, AU04_c, AU05_c, AU06_c, AU07_c, AU09_c, AU10_c, AU12_c, AU14_c, AU15_c, AU17_c, AU20_c, AU23_c, AU25_c, AU26_c, AU28_c, AU45_c
gaze_vel_X	Velocity of eye gaze along X-axis	$(\texttt{gaze\_angle\_x})'$
gaze_vel_Y	Velocity of eye gaze along Y-axis	$(\texttt{gaze\_angle\_y})'$
gaze_acc_X	Acceleration of eye gaze along X-axis	$(\texttt{gaze\_vel\_Y})'$
gaze_acc_Y	Acceleration of eye gaze along Y-axis	$(\texttt{gaze\_vel\_Y})'$
head_vel_T	Translational velocity of head	$\sqrt{(\texttt{pose\_Tx})'^2 + (\texttt{pose\_Ty})'^2 + (\texttt{pose\_Tz})'^2}$
head_vel_R	Rotational velocity of head	$\sqrt{(\texttt{pose\_Rx})'^2 + (\texttt{pose\_Ry})'^2 + (\texttt{pose\_Rz})'^2}$
head_acc_T	Translational acceleration of head	$\sqrt{(\texttt{pose\_Tx})''^2 + (\texttt{pose\_Ty})''^2 + (\texttt{pose\_Tz})''^2}$
head_acc_R	Rotational acceleration of head	$\sqrt{(\texttt{pose\_Rx})''^2 + (\texttt{pose\_Ry})''^2 + (\texttt{pose\_Rz})''^2}$
writing_speed	Speed of writing	–

TABLE I

FEATURES USED IN THE STUDY. X ARE FEATURES RETURNED BY OPENFACE. $(\texttt{X})'$ AND $(\texttt{X})''$ ARE THEIR FIRST AND SECOND DERIVATIVES.

and powerful supervised learning algorithms require substantial amounts of training data to learn reliable decision functions, application of affective computing is severely limited by short supply of trained expert annotators or potentially biased self-reports. To this end, we investigated the utility of an unsupervised representation learning model proposed by [30], which can be trained on a large amount of unlabeled data to learn potentially useful feature representations. By automatically learning useful features for classifying raw data, representation learning algorithms replace manual feature engineering and allow systems to identify potential discriminators and use them to support a specific predictive task. Very few studies have focused on unsupervised representation learning for time series and [30] is amongst the few general-purpose representation learning algorithms for time series without any structural assumptions on non-temporal data. Their model can learn representations from multivariate time series segments of varying lengths in a completely unsupervised fashion using a triplet loss function coupled with time-based negative sampling. The model (Figure 1) comprises of a deep neural network with dilated causal convolutions to handle time series [32]. This model minimizes an unsupervised triplet loss function which assigns similar time series proximate embeddings based on the assumption that they occur in temporal proximity while a distant subseries chosen at random (from either the same time series or a different one) is likely to be dissimilar. Therefore, for a reference time subseries x^{ref}, the paper chooses one of its own subseries as the positive example x^{pos} and another randomly chosen subseries x^{neg} as the negative example. In order to improve the convergence and the stability of the training procedure, the model chooses multiple negative samples independently. The training objective of the model is given by the following equation:

$$C = -log\Big(\sigma\Big(f\big(x^{ref},\theta\big)^T f\big(x^{pos},\theta\big)\Big)\Big) - \sum_{k=1}^{K} log\Big(\sigma\Big(f\big(x^{ref},\theta\big)^T f\big(x_k^{neg},\theta\big)\Big)\Big) \quad (1)$$

where $f(.,\theta)$ is a deep network with parameters θ and σ is the sigmoid function.

The unsupervised representation learning model was trained on 248 time series segments each having 27 features (refer Table I) over 300 time steps. The unsupervised model returns embeddings of a fixed and pre-determined shape. We trained our models for 4 different output dimensions of $(64, 1)$, $(128, 1)$, $(256, 1)$ and $(512, 1)$ respectively, and found that the model with 64 features performed comparably to more complex ones in the classification task, and we chose to use 64-dimensional embeddings as our featurization.

Using these output embeddings as feature vectors and manually annotated labels, we trained a random forest classifier to predict the cognitive state (Flow or Cognitive Disequilibrium) of a time series segment. We chose random forests because they are able to learn non-linear and complex decision boundaries, work well with high-dimensional data and can be robust to outliers. The unsupervised representation learning model coupled with a random forest classifier can function as a semi-supervised model, where the former learns embeddings (features) from a large number of time series segments in a completely unsupervised fashion, and the latter uses these features and a limited number of annotations to learn a decision function. Such a semi-supervised paradigm can be extremely useful in practice of affective computing, where obtaining vast amounts of unlabeled data is extremely easy, but its annotation can be expensive.

IV. RESULTS

A. Analysis of Time Series Complexity Measures

The results of K-S and randomization tests are illustrated in Table II. It can be clearly seen that the distributions of complexity measures of gaze velocity significantly differ across states of Cognitive Disequilibrium and Flow. Furthermore, distributions of Approximate and Sample Entropies of *all* behavioural features yielded significant differences between the two states. In order to investigate the directionality of the difference i.e. to answer whether behavioral signals in Cognitive Disequilibrium resulted in higher complexity than Flow or vice versa, we plotted the Empirical Cumulative Distribution Functions (ECDFs) for each complexity measure-feature pair which had a significant difference (Figure 2). The plots reveal that it is much more likely to observe lower

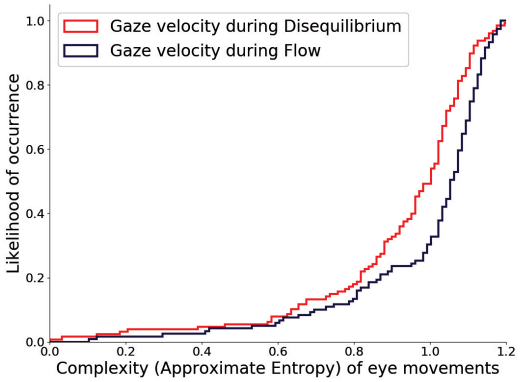


Fig. 2. ECDFs of Approximate Entropy (AppEn) values of Gaze velocity in CD (red) & Flow (blue). Given the AppEn of gaze velocity, it is much more likely to observe lower values of AppEn amongst CD than Flow.

complexity values in CD than in Flow. These results are in contrast to the findings of [18], which suggested that cognitive disequilibrium is correlated with a whitening of gross-body movement signals. Since whitening of a signal adds to its complexity, then gross body signals in Cognitive Disequilibrium should have higher complexity. However, our results (for instance Figure 2) consistently suggest otherwise. Inspired by these statistical results, we investigate the utility of complexity measures from a multivariate point of view in predicting CD and Flow.

B. Deep Feature Embedding Results

Figure 3 is a 3-dimensional UMAP [33] visualization of deep features (embeddings) returned by the unsupervised representation learning model. Subplots A and B represent embeddings of baseline non-personalized features and are colored by labels and subjects respectively. As shown, the deep features group the data points into two separate clusters

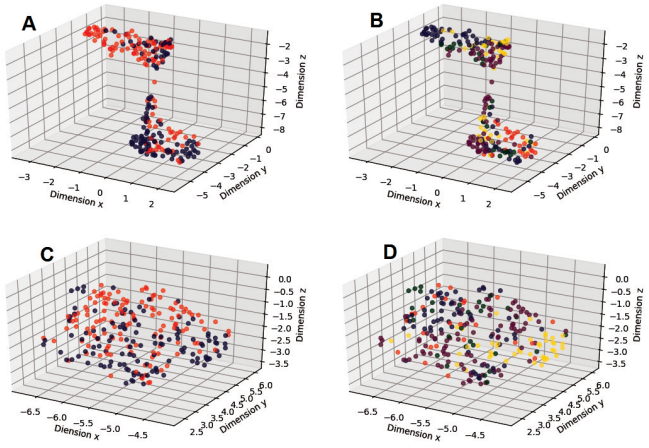


Fig. 3. 3D visualization of Deep Feature Embedding: (A) non-personalized features version colored by labels; (B) non-personalized features version colored by subjects; (C) personalized features version colored by labels; (D) personalized features version colored by subjects.

and in most cases the same subjects belongs to the same cluster. In other words, it seems that the deep embeddings have learned mostly the between-subject difference rather than the discrimination between labels. Subplots C and D are results from embedding learned from personalized features (i.e. features for a given subject are normalized using mean and standard deviation of the same subject aggregated across all sessions). As a result, the post-personalization features remove the between-subject variation and thus force the embeddings to learn something different, as can be seen from subplot D. It is however not obvious from subplot C whether the embedding is able to discriminate between labels due to its high dimensional feature space and possibly non-linear decision boundary, which motivates us to feed the embedding results into powerful classifier such as random forest for further evaluation. The next section presents results from these experiments.

C. Predictive Utility of Deep Features Embedding and Complexity Measures: Multivariate View

We conducted experiments to compare predictive utility of time series complexity measures and deep embedding features by feeding the two different features sets into random forest classifiers. We choose random forest for illustration as one a popular model type capable of learning complex non-linear decision boundaries. In order to test the utility of feature personalization/normalization, we compared the performance of the model using personalized and non-personalized features for both complexity measures and deep embedding features. In addition, we conducted three types of experiments given the hierarchical structure of the data: one subject has multiple sessions (one session is one child solving one problem) and one session has multiple time series segments. The first type of experiment ("Random") makes a *random split* between train and test sets, ignoring the grouping structures. This type of experiment could yield inflated algorithm performance as the information from the same session and the same subject may appear in both the training and testing sets, allowing the model to succeed by hooking-onto personal characteristics of some distinct subjects. The second type of experiment is conducted by *leaving one session out* ("LOSO") where the test set contains all data from one session (thus the same subject). This setup illustrates a "warm start" where we have data from all other subjects in addition to data from the same test subject, but from different sessions than the left-out test session. The last type is *leave-one person(subject)-out* ("LOPO"), which represents a "cold start" scenario where the model is trying to predict for a completely unseen subject. Due to varying degrees of information sharing between training and test set, we expect the performance will degrade from the upper bound case of random split, to LOSO and to the most conservative (but of most practical utility) LOPO experiments. Figure 4 shows the Area Under Receiver Operating Characteristic Curve (AUC) scores under various experimental conditions, comparing the effect of feature personalization and utility of deep embedding features versus

Complexity Measures	Gaze Velocity		Gaze Acceleration		Head Velocity		Head Acceleration	
	X	Y	X	Y	Translational	Rotational	Translational	Rotational
Approximate Entropy	**0.278**	**0.155**	**0.243**	**0.243**	**0.172**	**0.231**	**0.167**	**0.200**
Higuchi Fractal Dimension	**0.140**	**0.208**	0.119	0.119	0.122	**0.141**	0.071	0.114
Katz Fractal Dimension	**0.176**	**0.128**	0.085	0.085	**0.182**	**0.230**	**0.151**	**0.222**
Permutation Entropy	**0.215**	**0.218**	**0.283**	**0.283**	0.119	0.082	0.077	0.079
Sample Entropy	**0.259**	**0.166**	**0.178**	**0.178**	**0.177**	**0.224**	**0.190**	**0.210**
Spectral Entropy	**0.136**	**0.196**	**0.157**	**0.157**	**0.174**	**0.133**	0.119	**0.278**

TABLE II

KOLMOGOROV-SMIRNOV STATISTICS. VALUES IN **BOLD** INDICATE STATISTICALLY SIGNIFICANT DIFFERENCES AT 5% SIGNIFICANCE LEVELS. ALL THESE VALUES ALSO HAD EMPIRICAL p-VALUES < 0.05 RESULTING FROM THE RANDOMIZATION TEST. THE DISTRIBUTION OF COMPLEXITY MEASURES OF *gaze velocity* DIFFERS SIGNIFICANTLY ACROSS STATES OF COGNITIVE DISEQUILIBRIUM AND FLOW.

2*Features	Deep				Complexity			
	Non-personalized		Personalized		Non-personalized		Personalized	
Experiments	Random	LOPO	Random	LOPO	Random	LOPO	Random	LOPO
Precision	0.83 (0.037)	0.81 (0.063)	0.82 (0.035)	0.78 (0.083)	0.74 (0.062)	0.61 (0.099)	0.71 (0.030)	0.69 (0.143)
Recall	0.82 (0.04)	0.7 (0.111)	0.8 (0.050)	0.61 (0.159)	0.71 (0.070)	0.5 (0.138)	0.71 (0.033)	0.59 (0.143)
F1	0.82 (0.04)	0.71 (0.092)	0.8 (0.050)	0.61 (0.137)	0.71 (0.071)	0.48 (0.127)	0.71 (0.033)	0.60 (0.135)
Accuracy	0.82 (0.04)	0.7 (0.111)	0.8 (0.050)	0.61 (0.158)	0.71 (0.070)	0.5 (0.137)	0.71 (0.033)	0.59 (0.143)
AUC	0.83 (0.052)	0.79 (0.058)	0.8 (0.051)	0.74 (0.143)	0.72 (0.056)	0.43 (0.177)	0.71 (0.034)	0.55 (0.133)

TABLE III

PERFORMANCE COMPARISON OF DEEP FEATURE EMBEDDINGS VS. COMPLEXITY MEASURES, PERSONALIZED VS. NON-PERSONALIZED FEATURE SETS IN RANDOM AND LOPO EXPERIMENTS.

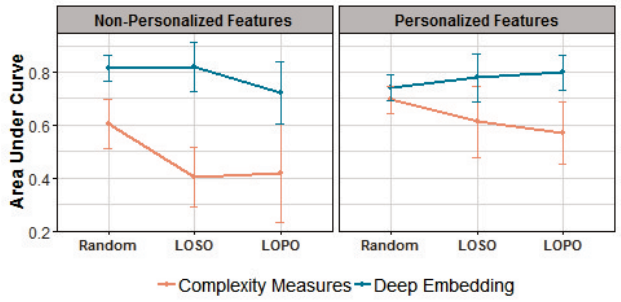


Fig. 4. Area Under ROC Curve (AUC) with 95% confidence interval, varying experimental conditions, feature penalization choices and featurization techniques (deep embedding vs. complexity measures).

baseline complexity features. The left panel shows the results from non-personalized features while right panel are those with personalized features. There are several interesting findings:

- *Effect of experiment conditions*: We observe a downward trend for both deep features and complexity measures from random split to Leave-One-Person-Out (LOPO, "cold start" condition), suggesting that the supervised model can be trapped to overfit on subject's specifics;
- *Effect of deep embeddings and complexity features*: As shown, deep features seem to have a clear advantage in predictive utility over complexity measures. This advantage is more prominent with non-personalized feature set;
- *Effect of feature personalization:* With complexity measures, personalization shows slight improvement from the non-personalized ones across all experiment conditions. With deep embedding features, it is interesting to note that the performance does not drop as significantly as in non-personalized version. In fact, the LOPO scores reveal a level of performance comparable with the random split evaluation with non-personalized features.

Table III presents detailed performance metrics (Precision, Recall, F1 score, Accuracy and Area Under ROC Curve) under different experiment conditions, generally consistent with data in Figure 4.

D. Towards Semi-supervised Learning: How Much Supervision is Necessary?

We conducted sensitivity analysis to demonstrate the utility of unsupervised embedding in the prediction task. In these set of experiments, we fixed the held-out test set, varied the size of the training set and reported the performance of our semi-supervised approach accordingly. For brevity, we only present results using non-personalized deep embedding features with random split and leave-one-person-out (LOPO) evaluation. As shown in Figure 5, *the model was able to achieve reasonable performance even with limited amount of supervision*. For random split, the performance drop is more prominent however within reasonable range. For the leave one person out (LOPO) condition, the performance is robust even with very limited amount of labeled data.

E. Comparison with Deep Supervised Learning

We also compared the performance of our semi-supervised model with *ResNet* [34]. [35] in a recent and comprehen-

sive survey found that ResNet can significantly outperform other deep learning approaches in classifying time series on the UCR/UEA and MTS archives. In addition, they found encouraging results (comparable predictive performance and significantly less training & testing time) while comparing ResNet to other state-of-the-art time series classification algorithms such as HIVE-COTE [36].

	Semi-supervised		Supervised (ResNet)	
Experiments	Random	LOPO	Random	LOPO
Precision	0.83 (0.037)	0.81 (0.063)	0.81 (0.016)	0.77 (0.074)
Recall	0.82 (0.04)	0.7 (0.111)	0.81 (0.023)	0.78 (0.071)
F1	0.82 (0.04)	0.71 (0.092)	0.81 (0.024)	0.77 (0.069)
Accuracy	0.82 (0.04)	0.7 (0.111)	0.81 (0.022)	0.78 (0.072)
AUC	0.83 (0.052)	0.79 (0.058)	0.8 (0.016)	0.73 (0.081)

TABLE IV

PERFORMANCE COMPARISON OF SEMI-SUPERVISED-MODEL & RESNET.

Table IV compares ResNet and the deep semi-supervised model introduced above on several performance metrics. For brevity, we only use the non-personalized feature set for two types of experiments: *random split* and *leave-one-person-out*. The results (Table IV) reveal that while ResNet achieved higher accuracy (0.78%) in the LOPO experiments, there was no significant difference (within 95% confidence interval) between the models in terms of AUC in both evaluation scenarios.

V. DISCUSSION

In this paper, we explored a semi-supervised framework to model the dynamics of involuntary behavioral signals collected using inexpensive sensors in order to discriminate between cognitive disequilibrium and flow as the primary input for decision making by human teachers or intelligent tutoring systems. Experimental results with a modestly sized multi-modal multi-sensor dataset, collected from young children practicing problem solving in a naturalistic environment, reveal several insights. Firstly, in comparison to time series complexity measures commonly cited in dynamical systems literature, we find that the deep feature embedding approach is able to identify plausible discriminators between those two states of interest more effectively than considered alternatives, when coupled with a random forest classifier. Secondly, we notice that this deep representation was able to effectively generalize from training subjects to previously unseen subjects, as demonstrated by its robust performance with leave-one-person-out experiments, and the advantage is even more pronounced with personalized features. Thirdly, sensitivity analysis with the semi-supervised framework shows that with deep embeddings features, the model is able to learn effective discrimination with even a small number of labeled data points, and the resulting performance is comparable with a potent fully supervised deep learning alternative which often requires large extents of supervision. When further validated with a more diverse set of subjects, the proposed approach has the promise to scale up practicality of the task of cognitive and affective state detection that is often bottle-necked by high costs of label acquisition even with abundant unlabeled data. Practically relevant capability of generalization to unseen subjects is also encouraging as the proposed approach would often be expected to work well with out-of-sample subjects in the real world use cases.

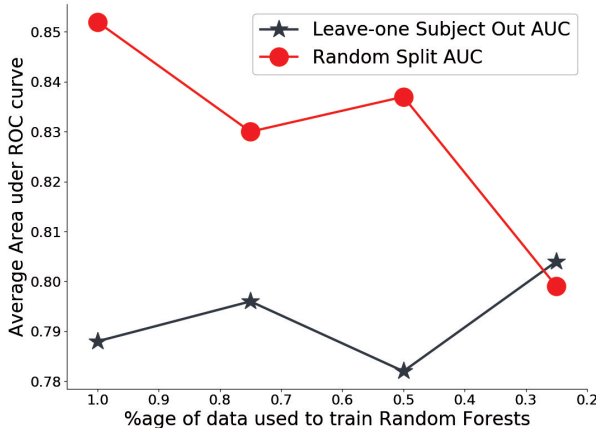


Fig. 5. Our semi-supervised model is able to achieve reasonable performance (AUC) even with limited amount of supervision

VI. CONCLUSION

Effective coaching of problem solving requires real time monitoring of students' cognitive and affective states, which can be challenging in societal environments with limited teaching resources. This paper tackles this challenge with a semi-supervised framework designed for automatic detection of two critical states of students during problem solving: Cognitive Disequilibrium and Flow. The discrimination model learns from involuntary behavioral signals that are less prone to social editing than more common alternatives, and that can be feasibly collected using inexpensive sensors. We empirically demonstrated the utility of the proposed approach and shown that it could work well even with a modest amount of data and limited supervision. When fully developed into a working system, we envision that the proposed methodology can play a role in augmenting human teacher's perceptual capability in the classroom as well as in improving the effectiveness of intelligent tutoring systems.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Ben Boecking and Xinyu Li in addition to other members of the Auton Lab and its interns (Chufan Gao, Harshayu Girase and Carl Edwards) for their useful advice on certain aspects of the experimental setup. This work was carried out when the first author was a Robotics Institute Summer Scholar in Carnegie Mellon University, USA. The first author would like to thank Ms. Rachel Burcin, Dr. John Dolan and Mikayla Trost for their support and encouragement throughout the summer.

REFERENCES

[1] L. Chen, X. Li, Z. Xia, Z. Song, L.-P. Morency, and A. Dubrawski, "Riding an emotional roller-coaster: A multimodal study of young child's math problem solving activities." *International Educational Data Mining Society*, 2016.

[2] M. Csikszentmihalyi, *Flow: The psychology of happiness*. Random House, 2013.

[3] S. D'Mello and A. C. Graesser, "Confusion," in *International handbook of emotions in education*. Routledge, 2014, pp. 299–320.

[4] B. J. Zimmerman and M. Martinez-Pons, "Student differences in self-regulated learning: Relating grade, sex, and giftedness to self-efficacy and strategy use." *Journal of educational Psychology*, vol. 82, no. 1, p. 51, 1990.

[5] S. D'Mello, "A selective meta-analysis on the relative incidence of discrete affective states during learning with technology." *Journal of Educational Psychology*, vol. 105, no. 4, p. 1082, 2013.

[6] A. Kapoor, W. Burleson, and R. W. Picard, "Automatic prediction of frustration," *International journal of human-computer studies*, vol. 65, no. 8, pp. 724–736, 2007.

[7] S. K. D'Mello and J. Kory, "A review and meta-analysis of multimodal affect detection systems," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 43, 2015.

[8] E. Joseph, "Engagement tracing: using response times to model student disengagement," *Artificial intelligence in education: Supporting learning through intelligent and socially informed technology*, vol. 125, p. 88, 2005.

[9] B. McDaniel, S. D'Mello, B. King, P. Chipman, K. Tapp, and A. Graesser, "Facial features for affective state detection in learning environments," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, no. 29, 2007.

[10] S. D'Mello and J. Kory, "Consistent but modest: a meta-analysis on unimodal and multimodal affect detection accuracies from 30 studies," in *Proceedings of the 14th ACM international conference on Multimodal interaction*. ACM, 2012, pp. 31–38.

[11] S. D'Mello and A. Graesser, "Autotutor and affective autotutor: Learning by talking with cognitively and emotionally intelligent computers that talk back," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 2, no. 4, p. 23, 2012.

[12] M. S. Hussain, O. AlZoubi, R. A. Calvo, and S. K. DMello, "Affect detection from multichannel physiology during learning sessions with autotutor," in *International Conference on Artificial Intelligence in Education*. Springer, 2011, pp. 131–138.

[13] L. A. Camras and J. M. Shutter, "Emotional facial expressions in infancy," *Emotion review*, vol. 2, no. 2, pp. 120–129, 2010.

[14] P. Ekman, "Strong evidence for universals in facial expressions: a reply to russell's mistaken critique." 1994.

[15] J. E. Kilbride and M. Yarczower, "Ethnic bias in the recognition of facial expressions," *Journal of Nonverbal Behavior*, vol. 8, no. 1, pp. 27–41, 1983.

[16] S. J. Heine, D. R. Lehman, K. Peng, and J. Greenholtz, "What's wrong with cross-cultural comparisons of subjective likert scales?: The reference-group effect." *Journal of personality and social psychology*, vol. 82, no. 6, p. 903, 2002.

[17] J. A. Krosnick, "Survey research," *Annual review of psychology*, vol. 50, no. 1, pp. 537–567, 1999.

[18] S. D'Mello, R. Dale, and A. Graesser, "Disequilibrium in the mind, disharmony in the body," *Cognition & emotion*, vol. 26, no. 2, pp. 362–374, 2012.

[19] M. Coulson, "Attributing emotion to static body postures: Recognition accuracy, confusions, and viewpoint dependence," *Journal of nonverbal behavior*, vol. 28, no. 2, pp. 117–139, 2004.

[20] S. DMello, T. Jackson, S. Craig, B. Morgan, P. Chipman, H. White, N. Person, B. Kort, R. el Kaliouby, R. Picard *et al.*, "Autotutor detects and responds to learners affective and cognitive states," in *Workshop on emotional and cognitive issues at the international conference on intelligent tutoring systems*, 2008, pp. 306–308.

[21] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "Openface: an open source facial behavior analysis toolkit," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–10.

[22] R. Esteller, G. Vachtsevanos, J. Echauz, and B. Litt, "A comparison of waveform fractal dimension algorithms," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 2, pp. 177–183, 2001.

[23] S. M. Pincus and A. L. Goldberger, "Physiological time-series analysis: what does regularity quantify?" *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 266, no. 4, pp. H1643–H1656, 1994.

[24] H. Kantz, J. Kurths, and G. Mayer-Kress, *Nonlinear analysis of physiological data*. Springer Science & Business Media, 2012.

[25] S. M. Pincus, "Approximate entropy as a measure of system complexity." *Proceedings of the National Academy of Sciences*, vol. 88, no. 6, pp. 2297–2301, 1991.

[26] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 278, no. 6, pp. H2039–H2049, 2000.

[27] G. Powell and I. Percival, "A spectral entropy method for distinguishing regular and irregular motion of hamiltonian systems," *Journal of Physics A: Mathematical and General*, vol. 12, no. 11, p. 2053, 1979.

[28] C. Bandt and B. Pompe, "Permutation entropy: a natural complexity measure for time series," *Physical review letters*, vol. 88, no. 17, p. 174102, 2002.

[29] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," *Physica D: Nonlinear Phenomena*, vol. 31, no. 2, pp. 277–283, 1988.

[30] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *arXiv preprint arXiv:1901.10738*, 2019.

[31] S. K. D'Mello, N. Bosch, and H. Chen, "Multimodal-multisensor affect detection," in *The Handbook of Multimodal-Multisensor Interfaces*. Association for Computing Machinery and Morgan & Claypool, 2018, pp. 167–202.

[32] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[33] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[36] J. Lines, S. Taylor, and A. Bagnall, "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 1041–1046.

Statistical Simulation for Multi-Agent Scheduling under Uncertainty

Quintessa Guengerich[1] and Isaac Isukapati[2]

***Abstract*— The Mat Sinking operation along the Mississippi River, which aims to protect the riverbed by covering it with concrete slabs, is currently being automated. Challenges arise in the automation of such a large-scale, dangerous task: with multiple large, robotic arms carrying heavy equipment over long distances, a scheduling algorithm is called for. However, scheduling the agents of the Mat Sinking robot is challenging, in part because task duration is dependent on safety, cost, and physical constraints. Uncertainties in these constraints creates a finite number of viable solutions in a vast search space. Furthermore, testing the performance of the scheduler often cannot be done on physical equipment, due to limited time and funds. In this paper, we present a statistical simulation of the Mat Sinking robot and its operating environment. The statistical simulator accounts for the uncertainty of the process, and allows for the testing of the scheduler without a physical prototype of the robot.**

I. MOTIVATION

Riverbeds suffer erosion due to sloughing and turbulence associated with water navigation. In order to prevent this erosion, a process called Mat Sinking is performed, in which concrete slabs are tied together into mats and lowered onto the riverbed. The concrete mat covers the riverbed, protecting from erosion and maintaining a safe waterway. To aid the process, the Mat Sinking group at the National Robotics Engineering Center is building a robot to lift the concrete slabs off of supply barges, tie them together, and lower them onto the riverbed.

Fig. 1. Current Mat Sinking operations [1]

However, scheduling the robotic arms of the robot to operate efficiently and safely is a challenge: scheduling algorithms have solution search spaces that increase in size combinatorially as new agents are added. In the case of mat sinking, this means that with each new robotic arm introduced to the system, the number of ways to perform the same task increases exponentially, with only a few suitable solutions. Furthermore, the testing of these scheduling algorithms often requires fully functional prototypes, that in turn require input from a scheduler to be optimally built.

Fig. 2. The Mat Sinking Robot in development at the National Robotics Engineering Center [2]

To address this gap, we have constructed a statistical simulation of the Mat Sinking process, allowing the scheduler to be tested quickly and realistically. The statistical simulator models the uncertainty of the process such as the time taken to move concrete slabs, and the likelihood of failure in each step of the process and tests the performance of the scheduler under different operating conditions.

In this paper, we first discuss the system architecture, illustrating the scheduler and simulator structures and their interactions in depth. Next, the experimental design for testing schedulers using the simulator is described. Finally, we present our expected results.

II. RELATED WORK

The Mat Sinking process can be considered a flexible job shop problem, where M jobs can be processed by any of N machines in a given set. Job shop scheduling is a well-studied and often applied problem, and simulation is frequently used to arrive at approximate solutions. [3] Optimization is often focused around minimizing makespan, or the time for the jobs to be complete, an optimization that is considered one of the most difficult NP-hard problems to solve. [4]

Statistical simulation, specifically, has also been applied to job shop scheduling problems. [5] In statistical simulation, parameters are estimated by sampling from distributions, thereby emulating the random and stochastic nature of many systems. In job shop scheduling, statistical simulation then renders the makespan as a cumulative distribution function.

[1]Quintessa Guengerich is a Chemical Engineering graduate from New Mexico Institue of Mining and Technology qguenger@gmail.com

[2] Isaac Isukapati is a staff scientist at Carnegie Mellon University

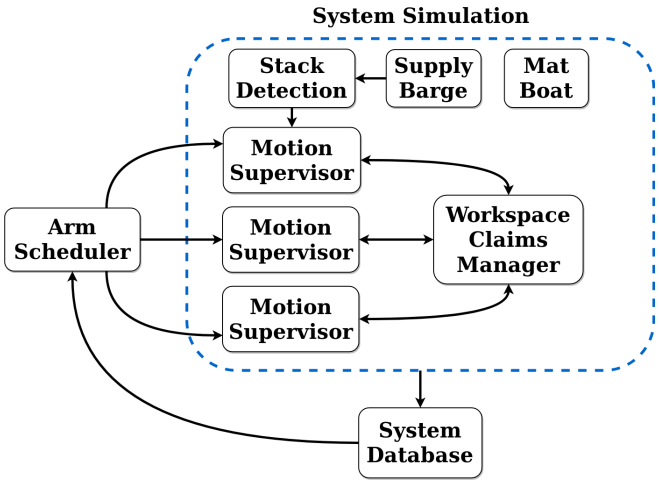


Fig. 3. Overall System Architecture, showing the components of the simulation

In Mat Sinking, the makespan should be minimized. In this paper, we explore two algorithms that allocate machines (lifting arms) differently to jobs (grasping and placing concrete slabs). We statistically simulate the makespan of each step along the way, and will compare the two schedulers by their maximum makespan.

III. BACKGROUND: MAT SINKING PROCESS

The Mat Sinking process is 4 basic steps:

1) Mat squares (with rough dimensions of 4 feet by 20 feet) are grasped and lifted off of a Supply Barge by up to 6 lifting arms, controlled by an Arm Motion Supervisor.
2) Mat squares are then moved from the Supply Barge to a Mat Boat, where they are placed in a line and tied together.
3) A launch is built when a certain number of squares are tied together – usually 35 squares. Once the launch is built, the launch is moved out of the way, and more launches are added to the mat.
4) When a mat is built, it's pushed into the water where it sinks to the bottom of the riverbed.

Ultimately, these tasks are aided by 6 robotic arms on a gantry.

Figure 4 shows a simplified schematic of the mat sinking environment. In the upper right-hand side of the image is the supply barge, with a bottom layer of stacks and a top layer of stacks. In the bottom left of the illustration is the row of mat squares, representing a single launch. The robot, which will interact with this environment, is not pictured here.

This environment, as well as the actions of the robot, are captured in the statistical simulation, discussed in the next sections.

IV. SIMULATION ARCHITECTURE

Figure 3 describes the overall system architecture of the simulation (outlined in blue) and the interacting components. The Arm Scheduler has access to data about the current state of the system, and uses that information to make decisions regarding each arm. Then, the scheduler passes high-level commands to the Arm Motion Supervisors, which control the movement of the robotic arms on a lower level in order to complete the task.

A. Arm Motion Supervisors

On the Mat Sinking robot and in simulation, the Arm Motion Supervisors (AMS) will control the movement of arms individually by issuing step-by-step commands, which include moving and sensing instructions to complete overall goals.

The AMS commands and the movements of the arm are simulated together, with methods that act on the Supply Barge and Mat Boat structures. These methods also estimate the success and failure of the task, as well as task duration, by randomly sampling from a distribution. The object accepts commands from the scheduler and sends requests to the Workspace Claims Manager. In order to complete tasks, the AMS must request space and receive permission from the Workspace Claims Manager, which checks that the requested space is not already claimed by another arm, discussed further below.

B. Supply Barge & Mat Boat

The supply barge and mat boats, illustrated in Figure 4, are simulated to be stacked on top of each other in a staggered configuration, with coordinates associated with each stack.

The availability of the bottom stacks depends on the availability of the top stacks: squares in a bottom stack are only available to be grasped by the robotic arm when the squares in the top stacks have been moved.

The Mat Sinking operation will indicate 23 stacks on the bottom and 22 stacks on the top layer, with each stack containing 12 squares. The lifting arm of the Mat Sinking robot lifts up to 2 squares at a time to deliver to the mat boat. The mat boat is simulated as a vector, which contains

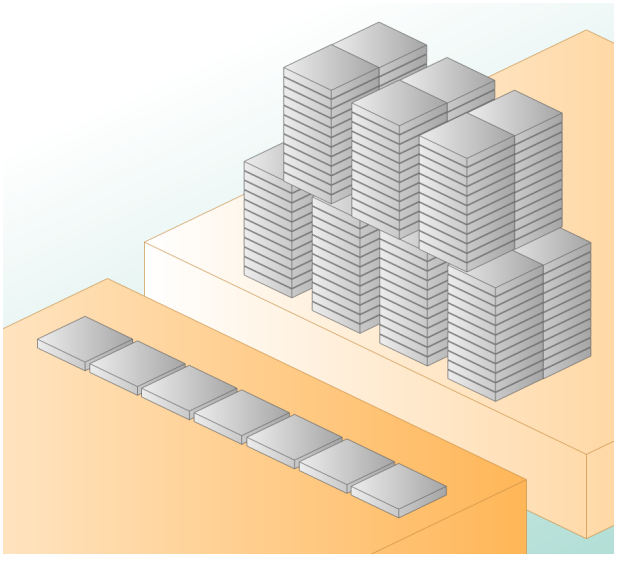


Fig. 4. Simplified illustration of supply barge and mat boat.

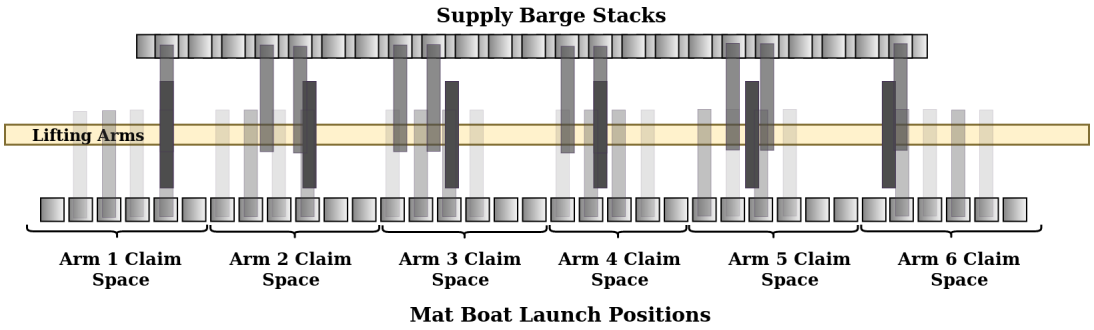


Fig. 5. In the basic scheduler, claimed space is preallocated for each arm, whose goal is to fill a specified fractions of the mat launch positions.

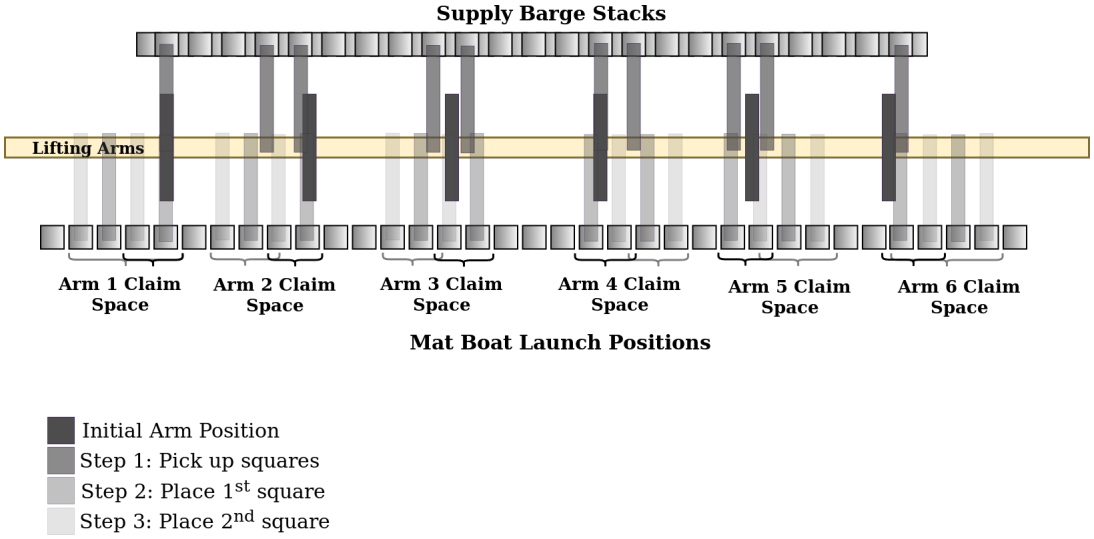


Fig. 6. In the Adaptive Scheduler, each arm is assigned to pick up stacks from the closest available stack, and place the stack on the closest available mat launch position, with claimed space allocated dynamically. In this way, an arm experiencing a high rate of failure can receive help from a neighboring arm to fill the nearby mat launch spaces.

up to 35 positions for squares. Each position has a coordinate associated with it, and is either empty or contains a square. Furthermore, space on both the supply barge and mat boat will be ”locked” if a lifting arm has claimed the space, preventing another arm from claiming the space.

C. Stack Detection

In order to grab and lift squares, each robotic arm must confirm its location above a stack on the supply barge by detecting a stack with a camera. In simulation, this is represented by randomly sampling from a distribution to generate estimated positions of the detected stacks and pass the point back to the AMS.

D. Workspace Claims Manager

The Workspace Claims Manager (WCM) evaluates a high level command received from the scheduler and evaluates the legality of the command *before* the arm completes the command. The WCM evaluates legality of a command by comparing:

- The current position of other arms
- The claim of the current arm
- The claim of other arms

What this means is that the WCM is preventing arms from attempting to fill the same space (ie, crashing into each other), by tracking their locations and their *intended* locations within the current command they are completing (their current claim.) If the WCM rejects a request for space, the AMS does not issue low level movement commands; instead, the AMS requests a new command from the scheduler.

V. METHOD

The statistical simulator is then used to test the performance of the scheduler. Two configurations are tested: a Basic Scheduler and an Adaptive Scheduler.

A. Basic Scheduler

In the basic scheduler – used as a control and ”worst case scenario” of efficiency – each arm is assigned a fraction of the stacks and a fraction of launch positions within their claim space, and they are then scheduled to fill those launch positions – a divide-and-conquer algorithm. In this case, the claimed space of each arm does not change. The illustration below in Figure 5 shows how these claim spaces could be distributed with six arms and 35 launch positions, with two steps and 4 squares placed.

B. Adaptive Scheduler

In the adaptive scheduler, each arm is assigned to grab squares from the closest supply barge stack, and place the squares on the closest mat launch positions, similar to a greedy algorithm. Figure 6 illustrates 2 steps of this process. Claimed space is allocated dynamically and freed when a task is complete, so that another arm can enter or move past the claimed space if necessary.

At first glance, the two schedulers appear to have similar results, and this is because the "nearest stack" depends largely on where the lifting arms are initialized on the gantry. However, we anticipate that the differences between the two schedulers will become more prominent under the following circumstances:

- when one launch of 35 squares is completely filled and moved, another launch must be filled, and the lifting arms begin the process in the position they left off in, which could be much closer together or further apart. This is only possible in the adaptive scheduler; in the basic scheduler, each arm will always be within it's declared claimed space.
- when some arms are functioning more slowly than others, or when arms are experiencing a higher rate of failure, other lifting arms will be able to enter the space that, in the basic scheduler, is claimed. This is only possible in the adaptive scheduler.

One exception exists in the algorithm: the rightmost and leftmost arms must have a preference to fill the ends of the mat launch, instead of the most nearby empty square, because only those arms can fulfill those requests.

VI. EXPERIMENTAL MATRIX

To compare the two schedulers, we will run them with the varied parameters shown in Table 1. Both the percentage failure and the number of functional (full speed, not slow or broken) arms are parameterized and will be varied to test the efficiency of the schedulers.

		Functional Arms				
		2	3	4	5	5
% Failure Expected	5	–	–	–	–	–
	10	–	–	–	–	–
	25	–	–	–	–	–
	50	–	–	–	–	–
	75	–	–	–	–	–

VII. FUTURE WORK EXPECTED RESULTS

These schedulers are still being implemented in C++ to interact with the simulation architecture discussed above. Upon running the simulation with the scheduler, the adaptive scheduler is expected to outperform the efficiency of the basic scheduler, most notably when slow arms are present in the system, because faster arms are less constrained. These faster arms will then be available to enter the space near the slow arms and aid in loading the launch area, which is not possible in the basic scheduler. Futhermore, when some arms are faced with a higher rate of failure and are stuck to fix or "retry" a task multiple times, arms facing lower rates of failure are free to move into nearby space to complete tasks on the supply barge and mat boat.

VIII. DISCUSSION AND CONCLUSION

Scheduling problems under uncertainty is a challenging problem. Discrete schedulers do not arrive at optimal solutions, and for real world problems with large uncertainty, a better solution is sought. In this paper, we discussed a statistical simulation of the Mat Sinking process along the Mississippi River, which is a process involving uncertainty on many fronts, such as inclement weather, wind, and conditions on boats. The statistical simulation described in this paper will be used to test two different schedulers, operating with different scheduling algorithms: divide-and-conquer and greedy. We will test both schedulers with different sets of constraints, such as percent failure expected and the number of functional arms, to illustrate a worst case scenario and a better solution, found via a statistical simulation.

ACKNOWLEDGMENTS

The author would like to thank:

- Dr. Isaac Isukupati
- Rachel Burcin
- Dr. John Dolan
- Traffic21 and Lisa Kay Schweyer

for their advice, guidance, and continued support of the project and the Robotics Institue Summer Scholars program.

REFERENCES

[1] Wired Staff, "Taming the Wild River." (2014) https://www.wired.com/2004/11/slideshow-taming-the-wild-river/

[2] O'Toole, Bill, "The ARMOR1, a massive floating factory, rises along the Allegheny at the NREC." (2019) https://www.nextpittsburgh.com/latest-news/the-armor1-a-massive-floating-factory-rises-along-the-allegheny-at-the-nrec/

[3] Ramasesh, R. (1990). Dynamic job shop scheduling: a survey of simulation research. Omega, 18(1), 43-57.

[4] Baker, C. T., Dzielinski, B. P. (1960). Simulation of a simplified job shop. Management science, 6(3), 311-323.

[5] Tavakkoli-Moghaddam, R., Daneshmand-Mehr, M. (2005). A computer simulation model for job shop scheduling problems minimizing makespan. Computers Industrial Engineering, 48(4), 811-823.

Imitation Learning for Latent Factors in Collaborative Multi-Agent Systems

Sharmistha Swasti Gupta[1], Dana Hughes[2] and Katia Sycara[2]

***Abstract*— Enabling robots to collaborate with humans requires both predicting the human's behavior, and determining appropriate robot responses. Imitation learning, or learning from demonstrations of human-human collaboration is a sample efficient way to achieve the same. Prior work on human-robot collaboration using imitation learning is mostly centered on the underlying assumption that human behaviours in a particular situation can be narrowed down to a specific expert policy. On the other hand, research that deals with varying human behaviour does not talk about scenarios where humans and robots collaborate without direct communication. In this paper, we propose a novel framework to train an agent that must learn to make intelligent observations and take sequential decisions in order to collaborate effectively with a human and work towards the same goal. We take into account the latent factors present in expert demonstrations and try to recover multiple expert policies. Thus, for every collaborator, an agent must be able to identify its type and also predict the best policy that it should adopt on its own, in order to have an effective collaboration.**

Multi-Agent Systems, Human-Robot Collaboration, Imitation Learning, Multiple Experts, Latent Factors

I. Introduction

The role of artificially intelligent systems in our lives is getting more prominent with time. There is now a preponderance of applications that require humans and agents to interact and cooperate with each other in order to complete a task. Thus, there is a need to develop robust frameworks that can facilitate effective human-robot teaming.

For robots to collaborate well with humans, they must be able to predict the human's behaviour accurately in addition to figuring out an appropriate response for that behaviour. Imitation learning, or learning from expert demonstrations, is a sample efficient approach to achieve the same. However, for a particular situation, human behaviours can vary significantly due to factors such as level of expertise, preference for a strategy etc. Additionally, a human might exhibit different behaviours under the same circumstance, at different points of time. This introduces unaccounted latent factors for an agent trying to learn from the human and essentially yields multiple distinct experts to learn from.

While [1] addresses this issue for a single agent, there does not exist a framework that enables multi-agent systems to distinguish between different experts. In multi-agent settings, there is an inherent relation between the agent responses: the optimal response of one agent can affect the response of the other agents. Thus, multiple solutions are possible in which, the agents' response might differ in each situation, but the collective goal is still achieved.

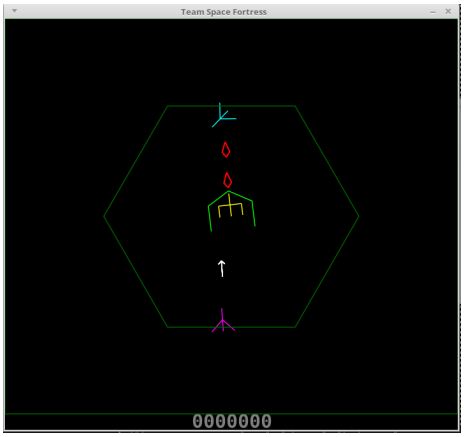


Fig. 1: Team Space Fortress Environment: The blue agent is the bait, which diverts the attention of the fortress, whereas the pink agent is the shooter, which attacks the fortress.

Team Space Fortress, shown in Fig. 1, is one such collaborative environment: two agents collaborate to destroy a fortress by taking on complementary roles of either a bait or a shooter. In a human-agent teaming scenario, the agent must be able to identify the type of human player from observations of the human's actions in the environment, and adopt a suitable policy for effective collaboration. However, the challenge with human behaviour data is that it is very difficult to confidently cluster them as separate types.

Through this work, we first propose a novel framework that aims to combine two existing adversarial imitation learning algorithms that can help learn latent factors like preferences of another agent in a multi-agent setting. Secondly, we propose environments where we can test this framework on self-generated data at present, so that it can later be extended to more complex human data.

The paper is structured as follows: Section II talks about prior work on collaboration in multi-agent systems using different approaches. Section III talks about the necessary background required to understand our proposed framework. In section IV, we introduce our approach. Section V describes our environments in detail and mentions the results and ongoing work. We finally conclude with discussions and future work in Section VI.

[1]Sharmistha Swasti Gupta is a senior undergraduate student in the Department of Electronics and Communications Engineering at Indraprastha Institute of Information Technology, New Delhi, India. sharmistha16193@iiitd.ac.in

[2]Dana Hughes and Katia Sycara are with the Advanced-Agent Robotics Technology Lab at the Robotics Institute in Carnegie Mellon University, Pittsburgh, USA. danahugh@andrew.cmu.edu, katia@cs.cmu.edu

II. Related Work

A. Learning in Collaborative or Competitive Multi-Agent Systems

Multi-agent cooperation and competition has been explored in literature using various deep reinforcement learning [2] and inverse reinforcement learning approaches [3] [4].
He et al. propose a framework in [5] to encode observations of the other agent into a deep Q-network (DQN) instead of predicting the action explicitly. There also exists literature on reinforcement learning with multiple experts [6] which employs a Bayesian Model Combination approach to learn a good combination of experts.
GIRL [7] is an inverse reinforcement learning based framework for a single agent through policy gradient minimization. Unhelkar et al. extend it to a multi-agent case by proposing a model-free method for the inverse reinforcement learning problem for a set of trajectories generated by different experts' policies [8]. However, algorithms like Q learning and policy gradient do not work for multi-agent systems because of non-stationarity of the environment in case of Q-learning and increase in variance with number of agents in case of policy gradient.
In [9], the authors propose a reinforcement learning framework independent of the number of agents or entities in the environment for cooperative behaviour in multi-agent teams where agents learn to cooperate by exchanging messages along the edges of a shared agent-entity graph. However, communication between multiple agents comes with a cost.

There also exist imitation learning approaches to multi-agent tasks. Zhan et al. [10] propose a generative multi-agent behavioral cloning framework that enables the agents to learn mappings from stages to distributions over multi-agent action spaces. However, the issue of generalization with behaviour cloning persists due to compounding errors and covariate shift [11] [12].
In [13], Song et al. propose a multi-agent generative adversarial learning framework that builds upon inverse reinforcement learning which can enable the agents to learn behaviours in cooperative or competitive high dimensional environments effectively. However, this work fails to take into account latent factors that might be present in expert demonstrations.

B. Learning Latent Factors in Expert Demonstrations

Springenberg et al. [14] propose a method to learn closely related expert policies by taking into account latent variables and exploiting a connection between reinforcement learning and variational inference. In [1] Song et al. develop an adversarial imitation learning framework for learning expert policies that are a function of a latent variable, using visual data as input. However, both these works are currently limited to single agent systems.

In [15], Unhelkar et al. attempt to learn the latent states in a human-agent collaborative multi-agent setting. However, their approach involves communication between the agents which is not efficient because communication, especially in sequential decision making, incurs cost [16].

III. Background

A. Markov Games

In a multi-agent learning domain, the Markov Decision Processes (MDPs) are generalized to Markov games. A Markov game [17] can be defined by a set of states, S and a collection of action sets, $A_1 ... A_k$, for each agent. Each agent uses a policy to move to the next state in accordance with a transition function, T: $S \times A_1 \times .... \times A_k \rightarrow P(S)$ where P(S) is the probability distribution of actions over the set S. Each agent also has a reward associated with it. The objective of each agent is to find a policy such that it maximizes its own total expected reward:

$$\sum_{j=0}^{\infty} \gamma^j r_{\text{t+j}}$$

where $\gamma \in [0,1)$ is a discount factor.

In a cooperative environment, in addition to the agent's own reward function, the total reward:

$$\sum_{i=0}^{k} \sum_{j=0}^{\infty} \gamma^j r_{\text{t+j}}$$

is also taken into consideration.

B. Imitation Learning

Imitation learning [18] uses expert demonstrations to learn a given task without knowledge of the reward functions. It learns a mapping between the current state and the demonstrated behaviour and attempts to optimize the error of deviating from that behaviour. It is efficient because it requires minimal expert knowledge of the tasks. Thus, it facilitates intelligent sequential decision making which is the major task at hand in teaming situations.

Inverse reinforcement learning is one of the approaches of imitation learning wherein data from expert demonstrations is used to infer the reward function of the agent. The agent then identifies the expert policy using reinforcement learning and adopts it. Fig. 2 shows the steps involved in an inverse reinforcement approach.

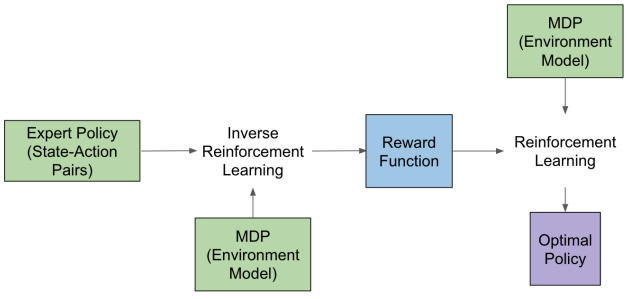


Fig. 2: Inverse Reinforcement Learning: Expert demonstrations are used to infer reward function. Then, reinforcement learning is used to determine optimal policy.

C. Generative Adversarial Imitation Learning (GAIL)

The GAIL framework [19] allows for directly recovering an expert policy from demonstration data as opposed to inverse reinforcement learning which recovers the reward function of the agent first. It uses a more expressive, non linear cost function thus proving to be more efficient than previous approaches. The cost function of GAIL is the negative log loss function of the binary classification problem of distinguishing between state action pairs of the generated policy and the expert policy [20].

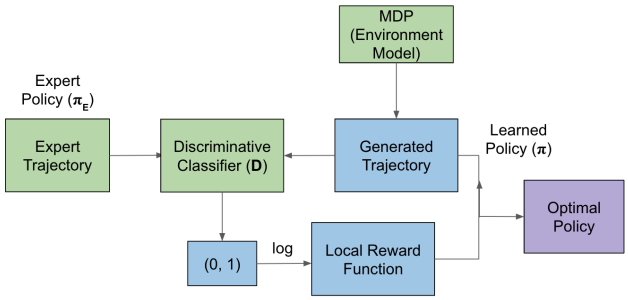


Fig. 3: Generative Adversarial Imitation Learning (GAIL): A discriminator tries to distinguish between trajectories generated by the learned policy and the expert policy.

As shown in Fig. 3, in GAIL, a policy is learned through a generative adversarial approach, where the policy is the generator function, and the discriminator tries to distinguish between trajectories of the learned policy, and expert demonstrations. The error of deviation of generated behaviour from expert behaviour is optimized by minimizing this error as measured by Jensen-Shannon divergence. It essentially finds the saddle point (π, D) of the expression:

$$E_{\pi}\Big[logD(s,a)\Big] + E_{\pi_E}\Big[log(1-D(s,a))\Big] - \lambda H(\pi)$$

where π is the generated policy, π_{E} is the expert policy, D is a discriminator which tries to distinguish state-action pairs from the trajectories generated by π and π_E, and

$$H(\pi) \triangleq E_{\pi}[-log\pi(a)]$$

is the γ-discounted casual entropy of the policy π_θ.

We consider two variations of GAIL:

1) MAGAIL: The MAGAIL framework [13] extends GAIL to multiple agents. It finds the saddle point (θ, ω) of the following function:

$$E_{\pi_\theta}\left[\sum_{i=1}^{N} logD_{\omega_i}(s,a_i)\right] + E_{\pi_E}\left[\sum_{i=1}^{N} log(1-D_{\omega_i}(s,a_i))\right]$$

where N is the number of agents, $\mathrm{D}_{\omega_{\mathrm{i}}}$ is the discriminator for an agent, which maps state-action pairs to scores optimized to distinguish between expert demonstrations and policy behaviours. π_{i}.

2) InfoGAIL: Data from human expert demonstrations show significant variability due to the presence of latent factors that result in multiple distinct expert policies. The InfoGAIL framework [1] infers the latency of such demonstrations in an unsupervised way. Let c denote a discrete latent variable that selects a policy π from the set of expert policies through p(π|c) (needs to be learned), where p(c) is the prior distribution of c (known before training). InfoGAIL recovers a policy π(a|s, c) close to the expert policy π_{E}; when c is sampled from the prior p(c), the trajectories (sequences of state-action pairs) τ generated by the conditional policy π(a|s, c) are similar to the expert trajectories τ_{E}, as measured by a discriminator.

If

$$L_I(\pi, Q) = E_{c(c),a\sim\pi(.,c)}[logQ(c|\tau)] + H(c)$$

where Q(c|τ) is an approximation of the true posterior P(c|τ). Then, the objective function of InfoGAIL is:

$$\min_{\pi,Q}\max_{D} \quad E_{\pi}\Big[logD(s,a)\Big] + E_{\pi_E}\Big[log(1-D(s,a))\Big] - \lambda_1 L_I(\pi(c),Q) - \lambda_2 H(\pi)$$

where $\lambda_1 > 0$ is the hyperparameter for information maximization regularization term, and $\lambda_2 > 0$ is the hyperparameter for the casual entropy term.

IV. Multi-Agent InfoGAIL

Fig. 4a) describes the framework for MAGAIL. Each agent has its own discriminator. However, there is no variable, as can be seen in the figure, that can account for latent factors present in the environment.

Fig. 4b) describes the framework for InfoGAIL. There is a discriminator which tries to get the generated policy as close as possible to the expert policy. The expert policy in this case is a function of a latent variable, c, which is an indicator or agent preference or expertise, depending on the environment.

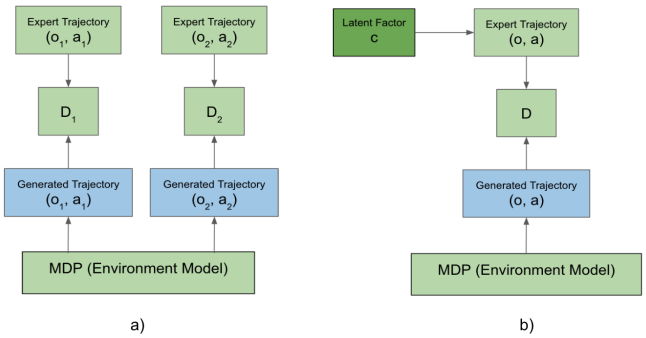


Fig. 4: a) MAGAIL framework: An extension of GAIL to multiple agents b) InfoGAIL framework: GAIL with an additional latent factor

Thus, to accomodate latent factors in a multi-agent setting, we propose a framework that combines MAGAIL and InfoGAIL, and call it Multi-Agent InfoGAIL.

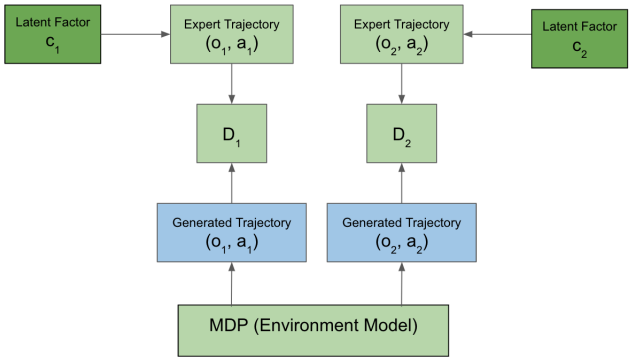


Fig. 5: Proposed Framework: A Combination of MAGAIL and InfoGAIL

As can be seen in Fig. 5, each agent has a discriminator. Additionally, there is a latent variable associated with the expert policy of each agent.

Thus, the resultant objective funtion of Multi-Agent Info-GAIL is to find the saddle point (θ, D) of:

$$E_{\pi_\theta}\left[\sum_{i=1}^{N} log D_{\omega_i}(s, a_i)\right] + E_{\pi_E}\left[\sum_{i=1}^{N} log(1 - D_{\omega_i}(s, a_i))\right]$$

$$- \lambda_1 L_I(\pi(c_1), Q) - \lambda_2 H(\pi(c_1)) - \lambda_1 L_I(\pi(c_2), Q)$$

$$- \lambda_2 H(\pi(c_2))$$

where $\lambda_1 > 0$ is a hyperparameter for the information maximization regularization terms and $\lambda_2 > 0$ is a hyperparameter for the casual entropy terms.

V. Environment Setup and Results

We set up two environments wherein the aim was to generate multiple expert trajectories, in order to test our approach.

Since the current Info-GAIL implementation uses visual data as input, we created an environment for position based input, which would eventually be useful in our TSF testbed. We modified OpenAI's CartPole environment [21] and added a position based bias to its reward function. Multiple such expert CartPole behaviours were generated which resulted in multiple expert trajectories.

For the multi-agent setting, we modified the existing simple spread scenario of OpenAI's multi-agent particle environment [22] to generate multiple expert trajectories of agent-agent teaming when covering certain landmarks.

A. Biased CartPole Environment (Single Agent)

The CartPole environment, as shown in Fig. 6 consists of a block that tries to balance a pole placed on top of it and gets a +1 reward for every successful step within the specified window: [-2.4, 2.4] on the horizontal axis. We introduce a bias by specifying a desired position of the block.

The additional position based term to the reward is then calculated as per the following Gaussian distribution (normalised to 1).

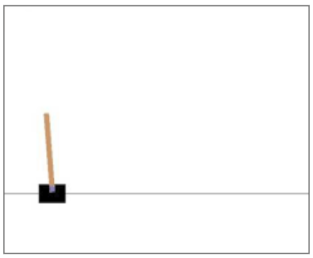


Fig. 6: Biased CartPole environment

$$P(x) = \frac{1}{\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

where x is the position of the block, μ is the mean (preferred) position of the block and σ is the standard deviation ("tolerance" to deviation from the position). Thus, the modified reward becomes 1 + $C * P(x)$ where C is a hyperparameter.

We generated multiple experts (standard deviation is taken as 0.0001 in all cases) using PPO (Proximal Policy Optimization) [23]. The expert agents differed in their preferences for the desired position along the x-axis.

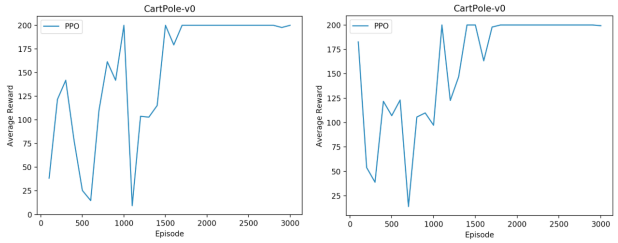


Fig. 7: Reward Function for Biased CartPole. Desired Position a) -2 b) +2

Fig. 7 shows the reward function of two such expert agents. Fig. 7a) indicates the plot of an agent with -2 as its desired position whereas Fig. 7b) indicates the plot of an agent with +2 as its desired position.

B. 4-Colored Tasks Environment (Multiple Agents)

There are 2 differently colored agents and 4 differently colored landmarks in the environment. Each agent has a preferred landmark color. The collaborative task of the agents is to cover a certain pair of landmarks, predefined to fetch the maximum reward. A bonus reward is received if an agent covers its preferred landmark.

In Fig. 8, let either of red (R) and green (G), or blue (B) and orange (O) be the pair that fetches maximum reward, if covered by the agents. Let us say the agents receive a reward of 10 each, if any of these pairs are covered. Let the preference of Agent 1 be red, and that of Agent 2 be orange. Let us say that the agents receive a bonus reward of 1 if they cover the landmark of their preference.

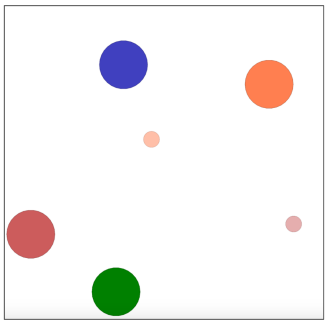


Fig. 8: 4-Colored Tasks Environment: The two agents must collaborate to cover the pair of landmarks that would fetch them maximum reward

Then, the payoff matrix of the agents would be as shown in Table I. The rows denote the landmark covered by Agent 1 (which prefers red) while the columns denote the landmark covered by Agent 2 (which prefers orange).
If the agents were to follow a game theoretical approach, and aim to arrive at a Nash equilibrium, their rewards would be (10, 10). However, since the task at hand focuses on effective teaming, the best option would be to get a reward of (11, 10) or (10, 11), which is what our approach tries to achieve.

Agent 1 \ Agent 2	R	G	B	O
R	(1, 0)	(11, 10)	(1, 0)	(1, 1)
G	(10, 10)	(0, 0)	(0, 0)	(0, 1)
B	(0, 0)	(0, 0)	(0, 0)	(10, 11)
O	(0, 0)	(0, 0)	(10, 10)	(0, 1)

TABLE I: Payoff Matrix: Cell values denote an ordered pair of rewards for Agent 1 (which prefers red) and Agent 2 (which prefers orange) on covering any of the R/G/B/O landmarks

The preference of an agent to move to a landmark of the same color is an indicator of the latent factors present in human decision making and this is what consequently yields clusters of multiple expert policies.

When a new agent is introduced into the environment, it must identify the type of the expert the other agent is, in this case, the color of the agent. It should then try to adopt a policy based on the cost function of the other agent so that it can collaborate with it in the best possible manner.

We are using MADDPG (Multi-Agent Deep Deterministic Policy Gradient) [24] to generate multiple experts such that the trained agent is able to continuously adapt to the preferences of the other agent.
The agent policy, as shown in Fig. 9 is defined such that it takes as input agent and landmark positions, takes into account its own preference, and also considers the preference of the other agent.

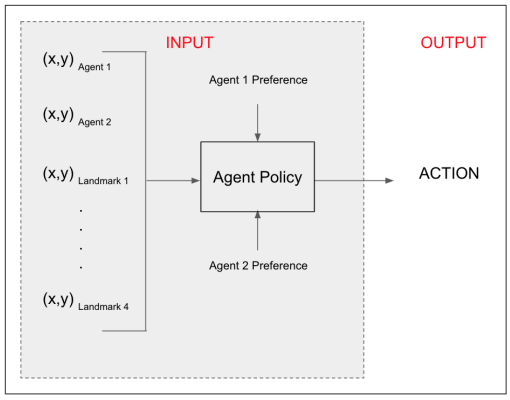


Fig. 9: Agent Policy: Preference of the other agent is also taken into account while taking action

VI. Discussion and Future Work

Our current work generates expert agents in two contexts. The single agent biased cartpole environment is a testbed for InfoGAIL with velocity, orientation and position as the input parameters, instead of visual data. This will enable us to establish confidence in the approach before we move to the Team Space Fortress environment which has similar data.

The multi-agent 4-Colored Tasks environment will be used to test our combined approach, Multi-Agent InfoGAIL, essentially learning latent factors in a collaborative multi-agent setting.

After we test our proposed framework on these simple environments, we will move on to a more challenging dataset of human expert demonstrations in a Team Space Fortress environment. Clustering human behaviours as different experts will prove to be a challenge. Another interesting research problem could be introducing a third agent in the environment and trying to establish effective collaboration.

Acknowledgment

The authors would like to thank members of the Advanced Agent-Robotics Technology Lab for their support. They would especially like to thank Swaminathan Gurumurthy and Sumit Kumar for their valuable input. They would also like to thank Rachel Burcin, Prof. John Dolan and the entire team of Robotics Institute Summer Scholars.

References

[1] Y. Li, J. Song, and S. Ermon, "Inferring the latent structure of human decision-making from raw visual inputs," *CoRR*, vol. abs/1703.08840, 2017. [Online]. Available: http://arxiv.org/abs/1703.08840

[2] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," *CoRR*, vol. abs/1812.11794, 2018. [Online]. Available: http://arxiv.org/abs/1812.11794

[3] T. S. Reddy, V. Gopikrishna, G. Zaruba, and M. Huber, "Inverse reinforcement learning for decentralized non-cooperative multiagent systems," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2012, pp. 1930–1935.

[4] L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 7194–7201. [Online]. Available: http://proceedings.mlr.press/v97/yu19e.html

[5] H. He, J. L. Boyd-Graber, K. Kwok, and H. D. III, "Opponent modeling in deep reinforcement learning," *CoRR*, vol. abs/1609.05559, 2016. [Online]. Available: http://arxiv.org/abs/1609.05559

[6] M. Gimelfarb, S. Sanner, and C.-G. Lee, "Reinforcement learning with multiple experts: A bayesian model combination approach," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 9528–9538. [Online]. Available: http://papers.nips.cc/paper/8162-reinforcement-learning-with-multiple-experts-a-bayesian-model-combination-approach.pdf

[7] M. Pirotta and M. Restelli, "Inverse reinforcement learning through policy gradient minimization," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 1993–1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016177

[8] D. Tateo, M. Pirotta, M. Restelli, and A. Bonarini, "Gradient-based minimization for multi-expert inverse reinforcement learning," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov 2017, pp. 1–8.

[9] A. Agarwal, S. Kumar, and K. P. Sycara, "Learning transferable cooperative behavior in multi-agent teams," *CoRR*, vol. abs/1906.01202, 2019. [Online]. Available: http://arxiv.org/abs/1906.01202

[10] E. Zhan, S. Zheng, Y. Yue, and P. Lucey, "Generative multi-agent behavioral cloning," 03 2018.

[11] S. Ross and D. Bagnell, "Efficient reductions for imitation learning." *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 661–668, 01 2010.

[12] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," *CoRR*, vol. abs/1011.0686, 2010. [Online]. Available: http://arxiv.org/abs/1011.0686

[13] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," *CoRR*, vol. abs/1807.09936, 2018. [Online]. Available: http://arxiv.org/abs/1807.09936

[14] J. T. Springenberg, K. Hausman, M. Riedmiller, N. Heess, and Z. Wang, "Learning an embedding space for transferable robot skills," 2018.

[15] V. V. Unhelkar and J. A. Shah, "Learning and communicating the latent states of human-machine collaboration," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, pp. 5789–5790. [Online]. Available: http://dl.acm.org/citation.cfm?id=3304652.3304855

[16] V. V. Unhelkar, X. J. Yang, and J. A. Shah, "Challenges for communication decision-making in sequential human-robot collaborative tasks," in *Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction at R: SS*, 2017.

[17] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds. San Francisco (CA): Morgan Kaufmann, 1994, pp. 157 – 163. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781558603356500271

[18] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 21:1–21:35, Apr. 2017. [Online]. Available: http://doi.acm.org/10.1145/3054912

[19] J. Ho and S. Ermon, "Generative adversarial imitation learning," *CoRR*, vol. abs/1606.03476, 2016. [Online]. Available: http://arxiv.org/abs/1606.03476

[20] "Generative adversarial imitation learning," https://slideplayer.com/slide/12976857/, last accessed on 12-August-2019.

[21] "Openai cartpole," url=https://gym.openai.com/envs/CartPole-v0/, last accessed on 21-Sept-2019.

[22] "Openai multi-agent particle environment," =https://github.com/openai/multiagent-particle-envs, last accessed on 21-Sept-2019.

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

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *CoRR*, vol. abs/1706.02275, 2017. [Online]. Available: http://arxiv.org/abs/1706.02275

Pre-computed Alternative Paths and Evaluation for Indoor Navigation Based on Elevation Information

Jiafan Hou[1], Zhaopei Gong[2] and Peng Yin[3]

***Abstract*— This paper describes the improvements of Pre-computed Alternative Paths (P-CAP) method for the application of indoor navigation. P-CAP was previously adopted to enable the aggressive aerial maneuvers in a fast selection and computation way. Since the method only checks the obstacles along the paths instead of searching or generating path online, it turns out to be a good candidate for safety and robust indoor navigation even in dynamic environment. The solution to the new application is separated into four steps. Various paths are pre-computed offline based on the fixed hypothetical map firstly. The planning environment is then online expressed as the grid map that store elevation information. By evaluating the traversability of each path, the velocity commands to track the selected path are finally sent out to the vehicle to perform obstacle avoidance and auto-navigation. The frame work was tested in simulator, and the result shows that the proposed method works for wheeled mobile robot in the normal indoor environment.**

I. Introduction

Intelligent unmanned vehicle is believed to have significant impact on human society. It can be applied for the construction of smart city, goods delivery and regular routing inspection in industry, etc. In recent years, researchers have put many efforts in this area and thanks to their great work, multiple methods and algorithms are developed and available for the auto-navigation with different purposes [1]. For instance, Genya, Keiji and Kazuya have developed a path planning algorithm for planetary exploration rovers [2]. Ji, Rushat, Vivek and Sanjiv defined P-CAP to enable aggressive aerial maneuvers in cluttered environments [3]. Systems like emergency braking with the help of active suspension [4], automatic parking [5] or blind angle vehicle detection [6] have been constructed for safer driving in urban environments.

With plenty of planning strategies though, still there are challenges in auto-navigation in dynamic environment due to the complicated real situations. Traditional searching based algorithms like A* or D*, etc. discrete the local environment to search optimal path, which lack efficiency and are time consuming for real time path planning to avoid moving obstacles. Learning and estimation based methods such as the ones described in [7] [8] [9], try to predict the objects motion paths and future locations with limited data base, which is not highly reliable and risky in application.

The strategy for auto-navigation introduced in this paper is based on the framework proposed by [3], who defines P-CAP and mainly focuses on the aggressive aerial maneuvers in cluttered environment. Autonomous flight in complex environments, as it figured out, is computationally expensive to plan and avoid obstacles because of the high speed and limited computation power. The paper thus declares the idea of generating paths offline beforehand, and selecting the one that is best suitable for the current situation to navigate towards the goal. Because of its fast speed in response to the environment for obstacle avoidance, it is believed that P-CAP is highly applicable for mobile robot in dynamics environment. However, in [3], online evaluation of pre-computed paths is limited in small available path space after checking obstacle collisions. The improvement of the method in this paper is therefore by constructing a local grid map that contains elevation information as an extra term for path evaluation. Since the test bed and concern of this paper focus on indoor ground navigation, the elevation information does make sense in path assessment. Evaluation based on this information is capable of reserving the whole path space and thus enhanced the method. Fig. 1 shows one situation for the path evaluation in local grid map in simulator, where the grids are constructed from the local environment in world frame and the axes are in vehicle frame.

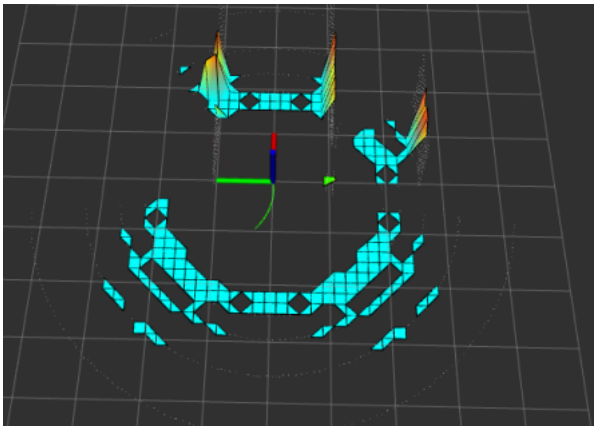


Fig. 1: Indoor environment expressed in grid map.

Furthermore, as the structure property of grid map permitted, it is also possible to add different layers of message into the grid map for higher level decision and strategy making in the future. Likewise, planning in grid map has

[1]Jiafan Hou is a senior student of School of Science and Engineering at Chinese Universiy of Hong Kong, Shenzhen, Shenzhen, China 116010072@link.cuhk.edu.cn

[2]Zhaopei Gong is a PhD. student of School of Mechatronics Engineering at Harbin Institute of Technology, Harbin, China gongzp@hit.edu.cn

[3]Peng Yin is with Biorobotics Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA pyin2@andrew.cmu.edu;

the potential to consider more difficult problems by adding layers of meaningful semantic information.

The structure of this paper is organized as follows. Section II proposes the frame work and the idea of navigation designed in details. The whole process is separated into four basic steps: 1) Pre-computation and generation of the path space. 2) Online grid map construction. 3) Path evaluation. 4) Path tracking. Section III discusses the conducted simulation and experiment results. Section IV eventually sums up the work and the future plan.

II. Frame Work

The solution to deal with indoor navigation based on elevation information was separated into four steps. A bunch of available path candidates are first generated offline in a basic map and stored to form a path space. The planning environment is then online constructed as grid map that stores elevation information and refreshed continuously. Based on the evaluation of traversability for each path in path space, the candidate that gains highest score with the consideration of heading angle, direction preference and obstacles avoidance is selected and fed into the path follower to perform auto-navigation. The frame work of the whole process is sum up to flow diagram shown in Fig. 2.

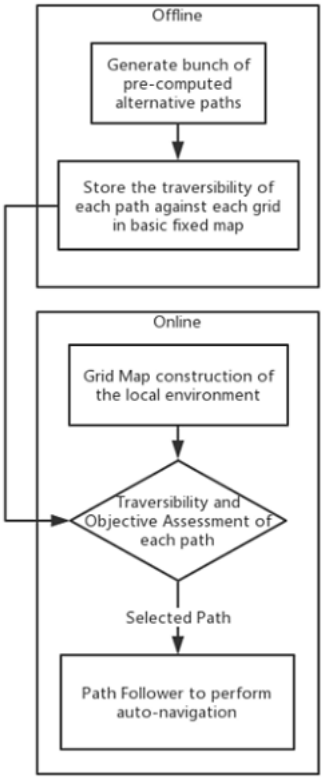


Fig. 2: Process flow for auto-navigation.

A. *Offline path space formation*

The local path candidates are generated before the online navigation. Thus, it is supposed that they cover most of the directions and are ready to be smoothly tracked with look ahead exploration. Based on these motivations, the paths are generated in the range of $[0, 2\pi]$ and each with two segments, one for navigation decision making and the other for heuristic exploration to avoid obstacles. The first part of the path is generated according to (1).

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} d \cdot \widetilde{x} \cos\lambda \\ d \cdot \widetilde{x} \sin\lambda \end{bmatrix} \tag{1}$$

where d is the planning path length, $\widetilde{x}$ is the sample space and λ is the switching angle of the path. Basically, the arcs of the circle that covers the search radius are grabbed and projected to the cartesian coordinates for smoothly navigation, illustrated by Fig. 3.

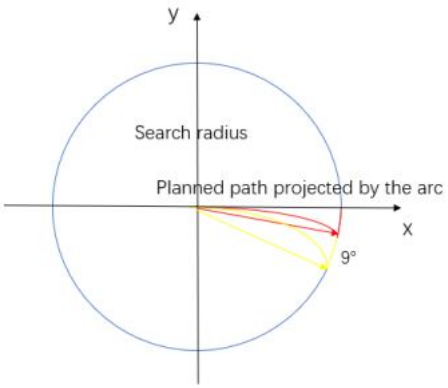


Fig. 3: Figure 3. Illustration of paths generated method.

This bunch of concave paths is then rotated in 2π to cover one round direction. The other part of the path is generated in the same way and brunches out from the start path for look ahead exploration. Fig. 4 demonstrates all the paths that cover the total area of $6 \times 6m^2$ generated by this idea ready for evaluation. Note that the start paths in the figure are in black spreading out from the right middle robot center, and the look ahead parts are colourful.

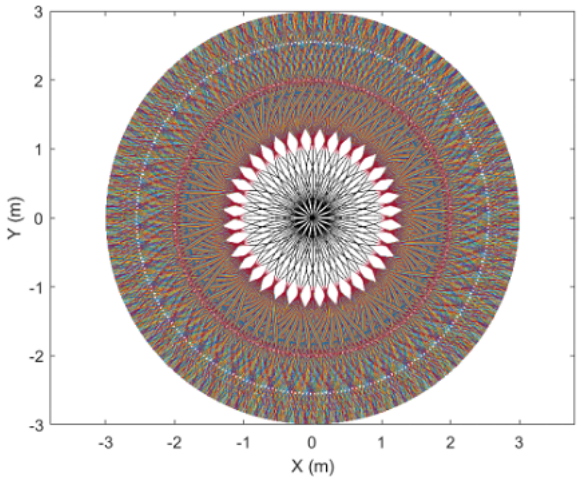


Fig. 4: All pre-computed path.

B. Robot-Centric Elevation Mapping

A robot-centric map is crucial for the mobile robot navigation in an unknown environment. In order to adapt the robot to a more complex indoor environment, online grid map that contains height information is constructed for path evaluation, whose process is named as elevation mapping for convention in this paper. In this section, we describe a Kalman-filter-based scheme for building elevation maps by integrating distance information from Lidar, whereas the map is incrementally build in real-time. The elevation map is represented by a two dimensional grid structure who stores the height of the grid h with variance σ_h^2. For the determination of the height for each location, the last points from the Lidar readings are transformed from robot relative distance measurements to global locations, with respect to the robots global pose as shown in Fig. 5.

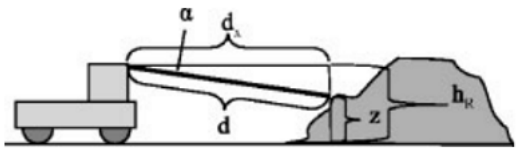


Fig. 5: Illustration of transforming range measurements to height values.

The measurements comes from the Lidar are first transformed into the robot-centric frame(x^r, y^r). The relative distance d_x and the height z of every measurement can be calculated as (2).

$$\begin{pmatrix} d_x \\ z \end{pmatrix} = F_{d\alpha} \begin{pmatrix} d \\ \alpha \end{pmatrix} = \begin{pmatrix} d\cos\alpha \\ h_R - d\sin\alpha \end{pmatrix} \tag{2}$$

where h_R is the height of the Lidar located on the robot. Then the map grid location can be calculated as (3)

$$\begin{pmatrix} x^r \\ y^r \end{pmatrix} = \begin{pmatrix} d_x\cos\beta \\ d_x\sin\beta \end{pmatrix} \tag{3}$$

where β denotes the horizontal angle of the laser beam.

Due to the inevitable noise caused by unknown environment such as the material and the light, the measurement would contain the noise. Thus, the height estimation is further updated from observation based on the history knowledge by applying Kalman filter(4),

$$\begin{cases} h(t) = \dfrac{1}{\sigma_{z_t}^2 + \sigma_{h(t-1)}^2}\left(\sigma_{z_t}^2 h(t-1) + \sigma_{h(t-1)}^2 z_t\right) \\ \sigma_{h(t)}^2 = \dfrac{1}{\frac{1}{\sigma_{h(t-1)}^2} + \frac{1}{\sigma_{z_t}^2}} \end{cases} \tag{4}$$

However, the mapping and updating method described in (4) is not suitable if the Lidar scans vertical structures such as walls. In that case, this method will lead to different height measurements and failed. The solution we figured out to walk out from this dilemma is restricting the application of mapping and updating it by introducing the Mahalanobis distance, which can be described as (5) [10]

$$h(t) = \begin{cases} z_t, \quad \text{if } z_t > h(t) \wedge d_M\left(z_t, h(t)\right) > c \\ h(t-1), \text{ if } z_t < h(t) \wedge d_M\left(z_t, h(t)\right) > c \\ h(t) = \frac{1}{\sigma_{z_t}^2 + \sigma_{h(t-1)}^2}\left(\sigma_{z_t}^2 h(t-1) + \sigma_{h(t-1)}^2 z_t\right), else \end{cases} \tag{5}$$

and variance $\sigma_{h(t)}^2$ with (6)

$$\sigma_{h(t)}^2 = \begin{cases} \sigma_{z_t'}^2, \text{ if } z_t > h(t) \wedge d_M\left(z_t, h(t)\right) > c \\ \sigma_{h(t-1)}^{z_t^2}, \text{ if } z_t < h(t) \wedge d_M\left(z_t, h(t)\right) > c \\ \frac{1}{\frac{1}{\sigma_{h(t-1)}^2} + \frac{1}{\sigma_{z_t}^2}}, \text{ else} \end{cases} \tag{6}$$

where d_M is the Mahalanobis distance defined as (7)

$$\mathrm{d_M}\left(\mathrm{z_t}, \hat{\mathrm{h}}(\mathrm{t})\right) = \sqrt{\frac{\left(\mathrm{z_t} - \hat{\mathrm{h}}(\mathrm{t})\right)}{\sigma_{\hat{\mathrm{h}}(\mathrm{t})}^2}} \tag{7}$$

Thus, a robot-centric elevation map can be obtained based on the Lidar measurement and the corresponding mapping procedure.

C. Path traversability evaluation and decision making

Now the grid map of local environment with appropriate resolution is ready for navigation decision making. Trading the online computation speed with memory allocation, the traversability for each path regarding every single grid is pre-memorized for online searching, as shown in Fig. 6. To

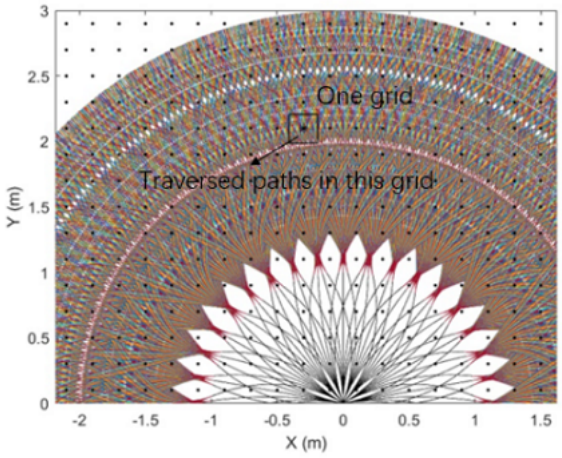


Fig. 6: Traversability for each path regarding every single grid.

select the best suited path in current local environment with look ahead distance, the score for each path is evaluated by the cost from the elevation, the preference direction and the desired heading angle, given by (8)

$$\text{score} = \left(1 - w_d \cdot \tilde{\theta}\right) \cdot \left(10 - \left|\frac{\theta - 180^\circ}{18}\right|\right)^3 - w_h \cdot \frac{\sum h_i}{r} \tag{8}$$

where in first term,

$$\tilde{\theta} = |\theta_{\text{desired}} - \theta_{\text{end of the path}}| \tag{9}$$

is the angle difference between the end point of the path and the desired heading direction (e.g. the global path direction), and w_d is the weight to normalize the difference. In second term, θ is the start point direction meaning the facing angle of each path. The term was included with the consideration that little turn would be preferred in planning. In last term, r is the distance from robot center to the current interested grid, $\sum h_i$ is the accumulated grid heights that the particular path pass by. In this way, lower score is given to the path that traverse grids with high elevation information and closed to the robot center, which is possibly blocked by the obstacles.

D. Path follower to perform tracking

After evaluating each path according to the constructed online grid map, the one with highest score that trading the heading angle, preference direction and looking ahead obstacles is selected. Classical PID controllers, who are the control strategy most frequently used in industry [11] are then applied for path tracking in terms of heading angle and velocity by (10) to perform auto-navigation.

$$\begin{bmatrix} \widetilde{\omega} \\ \widetilde{v} \end{bmatrix} = \begin{bmatrix} K_{p\omega}\delta_\vartheta + K_{i\omega}\int \delta_\vartheta + K_{d\omega}\frac{d\delta_\vartheta}{dt} \\ K_{pv}\delta_s + K_{tv}\int \delta_s + K_{dv}\frac{d\delta_s}{dt} \end{bmatrix} \tag{10}$$

where K_x are the tuning variables for each term and δ_θ and δ_s are the error terms to be controlled.

Simulation and Discussion

With the process described in Section III, simulation is first conducted in ROS base test bed in the virtual environment that recorded beforehand to test the method.

As the case shown in Fig. 1, whose original point cloud environment is the one shown in Fig. 7, the pre-computed path space is visited for the evaluation of current situation by (8), where the desired heading angle θ_{desired} for $\tilde{\theta}$ comes from joystick for testing. The green path shown in Fig. 8 gives out the final solution for tracking based on the objective function described by (8). With the move of the robot center, online grid map is refreshed according to the local environment in a certain range, and the assessment system eventually gives out the solution for continuous auto-navigation.

The simulation proves that the improved P-CAP can be applied in normal indoor navigation. But with its fast speed in path selection and evaluation, it is believed that the method can be also applied in dynamic environment for auto-navigation and obstacles avoidance. Further experiments are going to be conducted to show this capability to prove the claim.

Conclusion and Future Work

This paper proposes the new application of the improved P-CAP in indoor ground vehicle navigation with different kinds of the inputs. Basically, instead of reducing the path space due to the traversability against detected obstacles, the grid map structure that stores elevation information of the environment is adopted. Considering that particular information in path assessment, the entire path space is maintained for better performance of P-CAP. And with the fast speed property in path selection and decision making, improved P-CAP is believed to have the ability to navigate mobile robot in dynamic environment.

Simulation result shows that the method works for normal indoor environment. Experiment is going to be further conducted to prove the capability of its functioning in dynamic environment for auto-navigation. Last but not the least, as the structure of grid map permitted, it is also possible to add semantic information in the expression of different map layers to do higher level decision making for navigation in the future, which is going to be explored out in the next step.

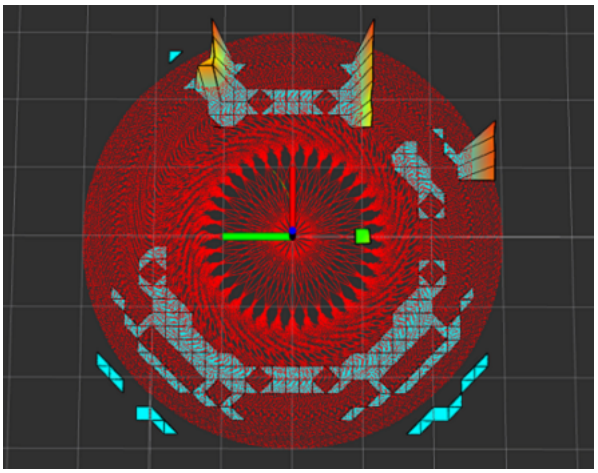


Fig. 7: Path evaluation in local grid map.

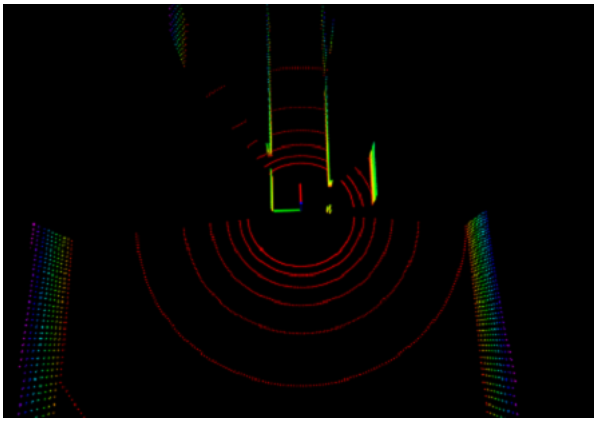


Fig. 8: Environment converted into grid map.

References

[1] D. Gonzlez, J. Prez, V. Milans, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, April 2016.

[2] G. Ishigami, K. Nagatani, and K. Yoshida, "Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 2361–2366.

[3] J. Zhang, R. G. Chadha, V. Velivela, and S. Singh, "P-cap: Pre-computed alternative paths to enable aggressive aerial maneuvers in cluttered environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8456–8463.

[4] J.-S. Lin and W.-E. Ting, "Nonlinear control design of anti-lock braking systems with assistance of active suspension," *IET control theory & applications*, vol. 1, no. 1, pp. 343–348, 2007.

[5] T.-h. Hsu, J.-F. Liu, P.-N. Yu, W.-S. Lee, and J.-S. Hsu, "Development of an automatic parking system for vehicle," in *2008 IEEE Vehicle Power and Propulsion Conference*. IEEE, 2008, pp. 1–6.

[6] R. O'malley, M. Glavin, and E. Jones, "Vision-based detection and tracking of vehicles to the rear with perspective correction in low-light conditions," *IET Intelligent Transport Systems*, vol. 5, no. 1, pp. 1–10, 2011.

[7] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1707–1725, 2010.

[8] A. F. Foka and P. E. Trahanias, "Probabilistic autonomous robot navigation in dynamic environments with human motion prediction," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 79–94, 2010.

[9] G. G. Yen and T. W. Hickey, "Reinforcement learning algorithms for robotic navigation in dynamic environments," *ISA transactions*, vol. 43, no. 2, pp. 217–230, 2004.

[10] A. Kleiner and C. Dornhege, "Real-time localization and elevation mapping within urban search and rescue scenarios," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 723–745, 2007.

[11] J. E. Normey-Rico, I. Alcalá, J. Gómez-Ortega, and E. F. Camacho, "Mobile robot path tracking using a robust pid controller," *Control Engineering Practice*, vol. 9, no. 11, pp. 1209–1214, 2001.

Enhancing Autonomy in Warehouse Truck Unloading Via Greedy Optimization

Ashton Larkin[1], Fahad Islam[2], Anirudh Vemula[2] and Maxim Likhachev[2]

***Abstract*— Warehouse applications typically involve manual labor to unload boxes from delivery trucks. This can be a time consuming task if a warehouse receives frequent shipments, and unloading boxes is a demanding task that can physically injure employees. In order to efficiently automate warehouse truck unloading, we modify the decision making components of an existing autonomous box unloading robotic system. The robot's goal is to maximize the throughput of boxes unloaded over a fixed time frame without dropping boxes or jamming the system. The physical interactions between the boxes in a truck create a stochastic environment, making truck unloading a complex task to automate using hand-designed strategies. We use machine learning techniques to learn task-level actions (e.g., *pick boxes from the top-right corner* or *scoop boxes from the bottom of the truck*). Training data is collected using a physics simulator, and self-supervision techniques are used to train our learning algorithm. Although the problem can be formulated as a partially observable Markov decision process (POMDP), the high dimensionality of the state space makes solving this problem with POMDPs intractable. Instead, we take a greedy approach and optimize the immediate reward by predicting the best action from a set of actions (using our trained classifier) for each scene of boxes that the robot encounters. Several experiments were run in simulation that tested our approach versus a planning under uncertainty approach. The results show that our method mimics the performance of planning under uncertainty in some environments and outperforms planning under uncertainty in other environments.**

I. INTRODUCTION

Unloading boxes from delivery trucks occurs frequently in warehouse settings. Automating truck unloading can help increase warehouse throughput and reduce employee work-related injuries. In order to automate warehouse truck unloading using a robot, the robot must be able to know what unloading action to perform given the state of boxes in a truck. The optimal action to perform is one that maximizes the number of boxes unloaded and minimizes the number of boxes dropped. The action the robot takes at a given state affects the next state of boxes the robot encounters, making this a sequential decision making process. Physical interactions between boxes also create a stochastic environment, which means that executing the same action on similar scenes may produce significantly different results.

The robot used in this work can be seen in fig. 1. The robot has an arm with suction cups that can be used for picking boxes, and a nose with conveyor belts that can be used for sweeping boxes. Example environments that the robot may encounter are shown in Fig. 2. The robot can execute an action like *pick boxes from the top-right corner of the truck*, or *scoop boxes from the bottom of the truck*. Since the robot should perform actions that maximize the number of boxes unloaded and minimize the number of boxes dropped, picking boxes closer to the top of the truck is best for the environments presented in Fig. 2(a) and 2(b), while scooping boxes from the bottom of the truck is best for 2(c) and 2(d). It's difficult to hand-design robust heuristics that enable the robot to make these correct decisions.

Fig. 1. The robot being used for autonomous truck unloading.[3]

There are many things to consider when deciding which action the robot should execute. The robot has a noisy perception system and also has no prior information about the boxes packed in a truck. The robot should not only avoid executing actions that drop a lot of boxes, but also avoid executing actions that unload too many boxes at once and end up jamming the unloading system. The boxes in a truck vary in size and mass, and box configurations depend on how they are packed. The variations between boxes and configurations create a large number of scenarios that the robot might encounter. Combining all of these factors makes automating warehouse truck unloading non-trivial.

In order to handle the stochastic aspect of this problem along with the uncertainties associated with the environments the robot encounters, a simulator can be used that mimics real world truck scenarios. The simulator can perfectly regenerate a given scene of boxes, allowing us to test each action on a particular scene to determine which action yields the optimal reward for that scene. This approach automatically generates correctly labeled data, which allows us to formulate the task of decision making as a self-supervised learning problem. After generating labeled data through self-supervision in

[1]Computer Science Department, Brigham Young University, Provo, UT 84602, USA adl95 at byu.edu

[2]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {fi, avemula1, mlikhach} at andrew.cmu.edu

[3]https://www.therobotreport.com/honeywell-robotic-unloaded-distribution/

(a) (b) (c) (d)

Fig. 2. Example environments that the robot may encounter at the start of truck unloading.

simulation, a multiclass classifier is trained to learn the relationship between box scenes and the optimal unloading action a robot should take.

Section II discusses the previous decision making approach used on the robotic system we are using, along with other approaches that may seem appropriate for this problem. Section III describes the problem formulation, including how self-supervised data collection and learning occurs. Section IV details the experiments that are conducted in order to compare our approach to the decision making approach taken in [1]. Section V shows the results obtained from conducting the experiments outlined in section IV and notes the advantages of our approach, along with future work to be done.

II. Related Work

A. Planning Under Uncertainty

Planning under uncertainty defines problem settings where uncertainty exists in the outcome of actions [2]. The truck unloading problem can be treated as planning under uncertainty since the environment is stochastic. The work done in [1] solves the truck unloading problem from this perspective by using a belief space planner [3] in an offline phase to generate strategies (i.e., a decision tree of states and actions) over a finite horizon. After these strategies are generated, online learning is used to choose the best strategy for the current scene. This approach is similar to the approach taken in [4]. The solution proposed in [1] is less complex than previous approaches that model planning under uncertainty as a partially observable Markov decision process [5]–[7].

B. Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning where an agent learns how to maximize some notion of cumulative reward in an environment through trial and error without human assistance [8]. RL has been shown to outperform humans in areas such as Atari games [9] by collecting large amounts of training data through fast simulators [10]. Since our simulator is slow, using RL makes it challenging for us to collect enough training data. Success has also been shown using RL in problem domains where training occurs slowly by running multiple agents in parallel and updating each agent's policy asynchronously [11], but the problems solved using this approach are not as complex as the truck unloading problem.

C. Self-Supervised Learning

Self-supervised learning is a type of supervised learning technique where the labelling of data is done automatically. Self-supervised learning can be used to solve problems that are otherwise unsolvable due to challenges in data collection [12]–[14]. In the context of our problem, we take a self-supervised learning approach since we have a simulator that can perfectly re-initialize scenes quickly. Since any scene can be re-initialized in simulation, we let the robot try every action on a given scene (when training) in order to determine which action gives the highest reward for a given scene. Repetitive experimentation on the same scene allows us to correctly label the scene, and we use the simulator to generate as many scenes as needed.

After generating various labeled scenes through simulation, the labeled scenes are combined to form a large data set that can be used to train a multiclass classifier. Although many self-supervised learning domains handle binary labels (e.g., object recognition) [15], we get continuous rewards for different actions. To account for this, we use weighted sampling, which is explained in section III-C.

III. Method

The approach we take in this work is aimed at helping the robot learn a mapping between a scene of boxes and the optimal action the robot should execute for that scene. We are doing this by training a classifier with data that was collected in simulation through greedy optimization. For this problem, greedy means that the robot only considers how to maximize the reward for the current scene it sees through an immediate action. The robot does not consider how the next scene will be affected by the current action. In this section, we will describe the problem formulation, data collection approach, and learning algorithm.

A. Problem Formulation

Assumptions:

- The collected training data is independent and identically distributed.
- The robot's perception system is perfect and gives us the ground truth about the box poses and dimensions.

The Robot. The robot has an arm with suction cups that can be used for picking boxes, and a nose with conveyor belts that can be used for scooping boxes. Refer to Fig. 1 for a depiction of the robot's arm and nose.

Robot Actions. There are six different pick actions and one sweep action that the robot can perform to unload boxes. These actions define an action set A, which is shown in eq. (1). When training, the robot tries every action in A on a given scene in order to determine which action $a \in A$ gives the maximum reward for that scene.

$$A = \{ \\ pickLeftLow\,,\, pickLeftMid\,,\, pickLeftHigh\,, \\ pickRightLow\,,\, pickRightMid\,,\, pickRightHigh\,, \\ sweep \\ \} \tag{1}$$

Rewards. For each scene the robot encounters, the goal is to choose an action that will maximize the reward, r. The reward is a function of the number of boxes unloaded, dropped, and jammed in the system for a given action. It's important to note that $r \in \mathbb{R}$ since there must be a penalty if boxes are dropped, or if the system gets jammed due to trying to unload too many boxes at once. The penalty for jamming the system is more severe than the penalty for dropping boxes because system jams require human intervention.

Algorithm 1: Data collection through self-supervision

```
Input: startScene, trainingIterations
Output: data
A = initializeActionSet();
data = new list();
iterationsRan = 0;
currScene = startScene;
while iterationsRan < trainingIterations do
    actionRewards = new list();
    nextAction = None;
    for a ∈ A do
        reward = currScene.performAction(a);
        actionRewards.append(reward);
    end
    if max(actionRewards) ≤ 0 then
        nextAction = A.defaultAction;
    else
        bestAction = argmax(actionRewards);
        nextAction = A.get(bestAction);
        features = extractFeatures(currScene);
        data.append(features, bestAction,
          actionRewards);
    end
    currScene = currScene.initNextScene(nextAction);
    iterationsRan += 1;
end
return data;
```

B. Data Collection

Feature Extraction. In order to save the simulation scenes with their corresponding best action, features must be extracted from the scene. In this work, feature extraction is motivated by the approach taken for the game of Tetris [16]. The features are defined by discretizing the scene into columns, saving the height of each column and the differences in heights between each column. This results in each scene being represented as a 95-dimensional feature vector. For an example of feature extraction, see Fig. 3.

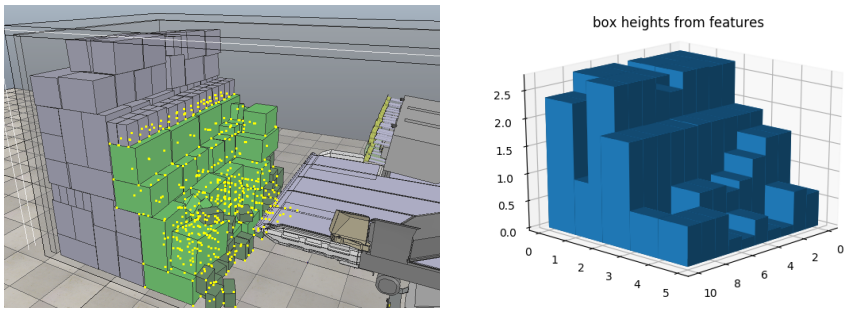
Data Collection Algorithm. The procedure for collecting training data is outlined in algorithm 1. In this algorithm, A is the action set defined by eq. (1). The data collection algorithm tries each action on the current scene, saves the best action and the rewards for that scene (if a best action exists), and then triggers the next scene by executing whatever action was defined as the best action for the current scene. It's important to note that all actions may produce a reward that is ≤ 0 for a given scene since there are penalties associated with dropping boxes and jamming the system. If this fail state ever occurs, we do not save this scene as part of the data set since there was no "optimal" action to pair it with. All we do for a fail state is trigger the next scene by executing a default action $d \in A$. In our experiments, we set d to $sweep$, but d can be set to any action $a \in A$.

Self-supervision. Another thing to note about algorithm 1 is that the labels for each scene are generated automatically through self-supervision. Since this algorithm is run on a simulator, we are able to try each action $a \in A$ on the same scene S. The simulator is key here because it allows us to reset S to its original state quickly whenever we want to execute another action on it. Resetting scenes manually instead of in simulation would not only take time, but would also void the guarantee that the scene was reset to the exact same state each time. By having a simulator, we can guarantee consistency and collect a large amount of correctly labeled data with ease.

C. Learning Algorithm

The Classifier. Once training data has been collected, this data is used to train a classifier whose goal is to learn a mapping between box scenes and optimal actions. Since there are more than two actions in A, we need to perform multiclass classification, not binary classification. We use scikit-learn's support vector classifier (SVM) as our multiclass classifier [17]. Details regarding the classifier's parameters are described in section IV-B.

Weighting the Training Data. When training this classifier, points are weighted based on the difference between the best and second best action rewards for a given scene. For some scenes, the optimal action produces a reward that is far better than all of the other action rewards. For other scenes, the optimal action reward is not significantly better than one or more of the other action rewards. We place a higher emphasis on fitting the classifier to scenes that have a



(a) The scene in simulation (b) The extracted features

Fig. 3. A visualization of how features are extracted from a particular scene.

distinctive best action to prevent the classifier from predicting actions that can potentially jam the system or achieve little to no reward. We place a lower emphasis on fitting the classifier to scenes that have multiple good actions because the penalty associated with predicting an action that performs close to the optimal action for a given scene is low.

Multi-Step Classification. For some scenes, it's difficult to distinguish the difference between performing a pick action on the left side of the truck vs the right side of the truck. Consider the scene depicted by fig. 2(a). Picking boxes from the top of the truck is the best action for this scene, but the side on which the high pick should occur is unclear. Since this is a stochastic environment, one experiment on this scene may come to the conclusion that *pickLeftHigh* is best, while another experiment on this scene may argue that *pickRightHigh* is best. This can limit the classifier's learning ability since it may receive multiple data points with very similar features, but different labels.

We avoid this issue by making action prediction a two step process. This involves relating the actions in A to a high level action set H, which is shown in eq. 2. The first step in this process is to determine which high level action $h \in H$ should be executed. This decision is made by the classifier. Once h is decided, the side on which h should be exectued must be determined if h is a pick action. The side is determined by extracting the total height of the boxes in the left and right halves of the truck from the features. If the total box height for the left half is greater than the total box height for the right half, then the pick action is executed on the left side. Otherwise, the pick action is executed on the right side. This decision making process is outlined in fig. 5.

$$H = \{pickLow\,,\, pickMid\,,\, pickHigh\,,\, sweep\} \qquad (2)$$

IV. Experiments

In order to compare the performance of our greedy optimization approach to the performance of the planning under uncertainty approach taken in [1], we train and test each approach on the same environments. Four simulation environments are used, shown in fig. 4. Environments A1 and A2 resemble neatly stacked boxes in a truck, while B1 and B2 represent messy boxes. Environments A1 and B1 are used to collect the training data. Environments A2 and B2 are used for evaluation. For information on how the training data is collected, see algorithm 1. V-REP [18] is the simulator used in this work, which is the same simulator used in [1].

A. Data Collection

Training data for this experiment was collected by running the simulator on environments A1 and B1. Ten simulations were run in parallel for each environment. Each parallel simulation ran for 50 iterations. Running parallel simulations reduces training time and also helps account for the stochasticity in the environments. After the runs were completed, the collected data from the parallel runs was combined into a single file. Any duplicate data points were removed during combination. A duplicate data point is defined as a point that has the same features and label. Duplicate points were likely to occur early in training since the parallel simulations for a given training environment started on the same scene. Data points that were correctly labelled but had rewards indicating that some of the actions failed to execute (due to either a system jam or communication timeout) were also removed. This resulted in 689 training data points.

B. Classifier Hyperparameter Tuning

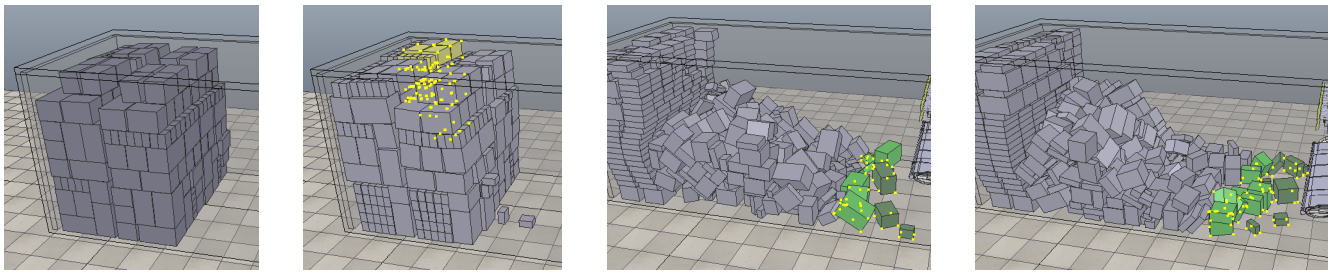
Once the training data was collected, the SVM hyperparameters were tuned using a grid search over the following parameters:

- Linear kernel
  - C: [1, 10, 100, 1000]
- RBF kernel
  - C: [1, 10, 100, 1000]
  - Gamma: [0.001, 0.0001]

Performing this grid search on the collected training data resulted in an optimal SVM having an RBF kernel with gamma set to 0.001 and C set to 1000.

C. Evaluation

Testing occurred on environments A2 and B2. Ten parallel simulations on each environment were run for evaluation. Before running the simulator on the test environments, the



(a) A1 (b) A2 (c) B1 (d) B2

Fig. 4. The simulation environments used for training and testing.

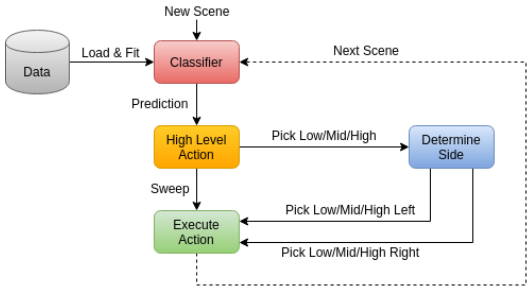


Fig. 5. The multi-step classification process.

classifier was fitted to the training data collected from environments A1 and B1. For testing, the multi-step classification process (see fig. 5) is used to determine what action to execute given the current scene. The robot then executes this action, which triggers the next scene that the robot encounters. This process is repeated until either all boxes have been unloaded from the truck, or the robot executes ten consecutive actions that result in no reward.

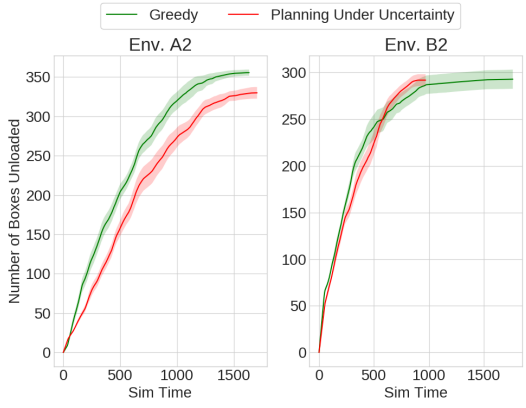


Fig. 6. The performance of our approach (green) against the approach outlined in [1] (red). The left subplot shows the evaluation results from environment A2, and the right subplot shows the evaluation results from environment B2.

V. Results and Discussion

The performance of our approach was compared against the performance achieved in [1] by performing the experiments described in section IV. The results of running these experiments are shown in fig. 6. Each line in fig. 6 represents the average performance of the 10 parallel runs, with the shaded area representing the standard error.

Our greedy approach outperforms [1] on environment A2 and performs similarly to [1] on environment B2. It's interesting to note that ignoring the effect the robot's current action has on the next scene does not reduce system performance. This is because the problem space for truck unloading is bounded by the walls of the truck. For problems with open environments like self driving cars, poorly made decisions at a given time step can lead to major error later on since the accumulated error in an open environment extrapolates more than the accumulated error in a closed environment. Our results confirm that the approach taken in [1] is overly complex, and that planning for extended horizons is unnecessary for this problem. Our approach is simpler than [1] and also performs no worse than the approach taken in [1].

A. Future Work

Multiclass Const-Sensitive Classification. In this work, we weighted the training data points based on the differences between the top two action rewards for a given scene. Weighting the training data does not make the classifier's loss proportional to the degree of error in the predictions made during training. We would like to define the classifier's loss as a function of the difference between the optimal action reward and predicted action reward.

Feature Engineering. We used features similar to the features used in [1]. Although our approach improved the robot's performance, these features do not seem to be informative for classification between task-level actions. Further work is needed to determine the proper feature representation for this problem.

VI. Acknowledgements

Thank you to Carnegie Mellon University's Robotics Institute For Summer Scholars (RISS) program and the National Science Foundation for supporting this work. Ashton would like to give special thanks to Michael Goodrich and the members of the Search Based Planning Laboratory at Carnegie Mellon University for their mentoring and guidance.

REFERENCES

[1] O. Salzman, A. Dornbush, F. Islam, S.-K. Kim, A. Vemula, and M. Likhachev, "Planning, learning and reasoning for robot truck unloading—a system's case study," *unpublished*, 2019. [Online]. Available: http://orensalzman.com/docs/ISRR19-Systems.pdf

[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[3] S. K. Kim, O. Salzman, and M. Likhachev, "POMHDP: Search-based belief space planning using multiple heuristics," in *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, July 2019.

[4] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 3338–3346.

[5] L. P. Kaelbling, M. L. Littman, and A. R.Cassandra, "Planning and acting in partially observable stochastic domains," *Artifical Intelligence*, vol. 101, pp. 99–134, May 1998.

[6] J. Pineau, G. Gordon, and S. Thrun, "Anytime Point-Based Approximations for Large POMDPs," *Journal Of Artificial Intelligence Research*, vol. 27, pp. 335–380, 2006.

[7] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online Planning Algorithms for POMDPs," *Journal Of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *NIPS Deep Learning Workshop*, December 2013.

[10] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2012.

[11] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[12] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3406–3413.

[13] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," *International Conference on Computer Vision*, 2017.

[14] B. Sofman, E. L. Ratliff, J. A. D. Bagnell, N. Vandapel, and A. T. Stentz, "Improving robot navigation through self-supervised online learning," in *Proceedings of Robotics: Science and Systems*, August 2006.

[15] Y. Wu, T. S. Huang, and K. Toyama, "Self-supervised learning for object recognition based on kernel discriminant-em algorithm," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, July 2001, pp. 275–280.

[16] C. Thiery and B. Scherrer, "Improvements on learning tetris with cross entropy," *International Computer Games Association Journal (ICGA)*, vol. 32, 2009.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

Signal Processing for Environment-Invariant WiFi Human Sensing

Yutian Lei[1], Fei Wang[2] and Dong Huang[3]

***Abstract*—With the high demand for wireless data traffic, WiFi networks have very rapid growth because they provide high throughput and are easy to deploy. Recently, Channel State Information (CSI) measured by WiFi networks is widely used for different sensing purposes. Though, there exists one particular challenge for the generalization of WiFi sensing, that the current algorithms are not robust enough to deal with the varied environments. In this paper, we present the methods to remove the random phase shift of Channel State Information (CSI) by estimating sampling time offset (STO), sampling frequency offset (SFO) and Cyclic Shift Diversity (CSD). Also, the (MUSIC) MUltiple SIgnal Classification algorithm is discussed to estimate multi-path features of signal propagation. In addition, we propose a novel way to calibrate the relative position of the transmit and receive antennas, and make the CSI less sensitive to the location and orientation of the antennas. Our experiments shows the modified CSI are to some extent free from variance of the random noise from the environments and hardware.**

***Index Terms*—MUSIC, signal processing, phase removal, WiFi, CSI**

I. Introduction

Recent advances in WiFi technology have accelerated the popularization of wireless devices. Due to the widespread deployment of wireless networks, WiFi sensing with Channel State Information (CSI) provided by Multiple-Input Multiple-Output (MIMO) is actively researched these years. CSI describes how a signal propagates from the transmitter to the receiver at certain sub-carrier frequencies and represents the combined effect of, for example, scattering, fading, and power decay with distance. A time series of CSI data can actually capture the spatial and movement information of the environment, and thus can be used for different wireless sensing applications, like human detection [1], indoor localization [2]–[4], gesture recognition [5], activity recognition [6], [7].

Due to the multi-path effect, the raw CSI data is the superposition of signals in different paths, which contains redundant information from the static environment. In addition, the CSI measurements are sensitive to the location and orientation of the antennas, meaning even recent research about WiFi sensing requires the transmit and receive antennas to be placed in fixed positions. To some extend, it restricts the real-world application of environment-invariant WiFi sensing since it's difficult to duplicate the same setting of the transmit and receive antenna.

Therefore, signal processing methods for CSI data are explored in this paper to make the CSI capable of environment-invariant sensing. The method consists of three parts: 1) Phase Offsets Removal: removing the phase offsets due to hardware and software errors like Sampling Time Offset (STO), Cyclic Shift Diversity (CSD) and Sampling Frequency Offset (SFO); 2) Muiltipath Feature Estimation: Estimation the feature of interest of different paths by MUSIC(MUltiple SIgnal Classification)-based algorithm; 3) Signal Calibration: Calibrating the relative position of the receiver and transmitter.

The paper is organized as follows: section II introduces the preliminary of Channel State Information (CSI), section III explains the methods and reasons to remove random phase shift, section IV shows MUSIC algorithm for estimating the multi-path feature, section V introduces the novel way to make signal calibration and VI covers the experiments and results.

II. Preliminary: Channel State Information

CSI characterizes how wireless signals propagate from the transmitter to the receiver at certain carrier frequencies. CSI amplitude and phase are impacted by multi-path effects, including amplitude attenuation and phase shift. Each CSI entry represents the Channel Frequency Response (CFR),

$$H(t;f) = \sum_{l}^{L} \gamma_l(t) e^{-j2\pi f \tau_l(t)} \tag{1}$$

where f is the carrier frequency, $\gamma_n(t)$ is the amplitude attenuation factor and $\tau_n(t)$ is the propagation delay for path l. [8] The CSI amplitude $|H|$ and phase $\angle H$ are impacted by the displacements and movements of the transmitter, receiver, and surrounding objects and humans. In other words, CSI captures the wireless characteristics of the nearby environment. These characteristics can be utilized by mathematical modeling or machine-learning algorithms for different sensing applications. This is the rationale for why CSI can be used for WiFi sensing. [9]

For a Wifi System with N transmit antennas, M receive antennas, and K subcarriers operating for T time, the CSI data is a 4D matrix with shape $M \times N \times K \times T$.

III. Phase Offsets Removal

For real-world commodity Wi-Fi devices, expect the multi-path effect, the measured CSI is also impacted by many practical hardware and software problems. First, the receivers and the transmitter are not tightly time synchronized, so their sampling clocks at the DAC and the ADC are not in sync, which causes sampling time offset (STO). Apart from sampling time offset, there is also a sampling frequency offset (SFO) between every WiFi sender receiver pair. SFO changes the sampling time offset from packet to packet for the same sender-receiver pair [4]. In addition, the Orthogonal Frequency-Division Multiplexing (OFDM) causes Cyclic Shift Diversity (CSD).

[1]Yutian Lei is a senior student of Computer Science and Engineering Department at The Chinese University of Hong Kong, Shenzhen, Shenzhen 518116, China. 115020250@link.cuhk.edu.cn

[2,3]Fei Wang, Dong Huang are Delight Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA feiw2@andrew.cmu.edu; dghuang@andrew.cmu.edu

Fig. 1. WiFi antennas as sensors for human sensing

Taking these into consideration, the measured CSI is therefore [9]:

$$H_{m,n,k} = (\sum_{l}^{L} \gamma_l e^{-j2\pi f_k \tau_{m,n,l}}) e^{-j2\pi a_m f_k} e^{-j2\pi b f_k} e^{-j2\pi c(f'_k - f_k)} \tag{2}$$

where $H_{m,n,k}(t)$ is the CSI from receiver i to transmitter j at subcarrier k, a_m is the time delay from Cyclic Shift Diversity (CSD) of the nth transmit antenna, b is the Sampling Time Offset (STO), c is the Sampling Frequency Offset (SFO).

According to (2), the phase of measured CSI is

$$\angle H_{m,n,k} = \Phi_{m,n,k} - 2\pi f_\sigma k(a_i + b + c(f'_k/f_k - 1)) \tag{3}$$

The phase offsets are estimated by minimizing the linear fitting errors across K subcarriers, N transmit antennas, and M receive antennas.

$$\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_0 = arg \min_{\beta_1,\beta_2,\beta_0} \sum_{i,j,k} (\angle H_{i,j,k} + \beta_1 ki + \beta_2 k + \beta_0) \tag{4}$$

where $\beta_1, \beta_2, \beta_0$ are the fitting variables. From the unwrapped CSI phase, subtract the phase that would have been added due to STO, SFO and CSD to obtain modified CSI phase $\angle \hat{H_{i,j,k}}$ as

$$\angle \hat{H_{i,j,k}} = \angle H_{i,j,k} + \beta_1 ki + \beta_2 k + \beta_0 \tag{5}$$

IV. Muiltipath Feature Estimation

Fig 2 shows a simplified propagation model for WiFi signals, which consists of three types of path: 1) The static reflection paths caused by the static objects in the environments; 2) The direct path between the transmitters and receivers. (Note that the direct path may not exist if there are obstacles between the transmitters and receivers); 3) The moving paths caused by human movement. The propagation paths of WiFi signal can characterized by multi-dimensional features. And in this section, we will introduce what are these features, how can they affect the CSI and how to make estimations of these features.

A. Feature of Propagation Paths

Angle of Arrival and Angle of Departure The angle of arrivals (AoAs) θ_l is defined as the angle between the normal of the receiver antenna and the direction of the signal arriving at the receiver for path l, and the angle of departures (AoDs) ω_l is defined as the angle between the normal of the transmit antenna and the direction of the signal sending at the transmitter for path l.

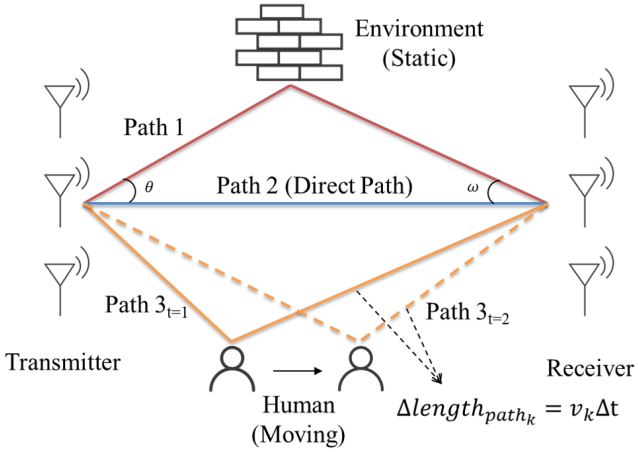


Fig. 2. Multi-paths of the Signal Propagation

Assume there are M receiver antennas and N transmit antennas in a uniform linear array, and wireless channel is far-field (FF) narrowband. Let's define L as the number of propagation paths, and d_R be the distance between consecutive two receivers antennas, so that the difference of path length between two receivers for path l will be $d \sin\theta_l$. Let's denote

$$\Phi(\theta_l) = e^{-j*2\pi*d_R*sin(\theta_l)*f/c} \tag{6}$$

where c is the speed of light and f is the frequency of the transmitted signal. So relative to the first receiver antenna in the array, the phase shift introduced at m^{th} antenna of path l will be $\Phi^{m-1}(\theta_l)$

Similarly, let's denote

$$\Omega(\omega_l) = e^{-j*2\pi*d_T*sin(\omega_l)*f/c} \tag{7}$$

where d_T be the distance between consecutive two transmit antennas. So relative to the first transmit antenna in the array, the phase shift introduced at n^{th} antenna of path l will be $\Omega^{n-1}(\omega_l)$.

Time of Flight The time of flight (ToF) t_l is defined as the propagation time the signal takes along a particular path. Assume there are K subcarries and the frequency difference between consecutive subcarries is f_δ. The subcarries with

different frequencies have different wavelengths, and thus introduce phase shifts. Let's denote

$$T(t_l) = e^{-j*2\pi*d*f_\delta*t_l} \tag{8}$$

So relative to the first subcarrier, the phase shift introduced at k^{th} antenna of path l will be $T^{k-1}(t_l)$.

Doppler velocity The Doppler velocity v_l is defined as the change speed of the path length due to the human movement. It introduce phase shift at consecutive sampling time for the moving path l. Let's denote

$$V(v_l) = e^{-j*2\pi*d*t_\delta*v_l/c} \tag{9}$$

where t_δ is the sampling interval. So relative to the data of time 0, the phase shift introduced at t^{th} time of path l by Doppler shift will be $V^{t-1}(v_l)$ [10].

B. Estimation of the Features

MUSIC (MUltiple SIgnal Classification) algorithm is first proposed to estimate the Angle of Arrivals (AoAs) θ_k of the propagation path [11]. Recently, there are many variants of MUSIC-based algorithm developed for estimating other features of interest of different propagation path, like Time of Flights (Tofs), Angle of Departure (AoDs) and Doppler velocity. To gain insight into how the variants of MUSIC-based algorithm works, it is helpful to understand how standard AoA computation with the well known MUSIC algorithm works, which we review next.

According to Eq 6 and Eq 1, and considering only the phase shift caused by AoAs, the CSI relative to first receiver will be

$$H_m(t;f) = \sum_l^L \gamma_l(t) e^{-j2\pi f \tau_0} \Phi^{m-1}(\theta_l) = \sum_l^L \gamma_l'(t) e^{-j2\pi f \tau_0} \tag{10}$$

let $\alpha(\vec{\theta_l}) = \begin{bmatrix}1 & Phi(\theta_l) \cdots Phi^{M-1}(\theta_l)\end{bmatrix}$. This vector $\alpha(\vec{\theta_l})$ is also known as the steering vector. We have as many steering vectors as propagation paths and the overall steering matrix A is defined as $A = \left[\alpha(\vec{\theta}_1) \quad \alpha(\vec{\theta}_2) \cdots \alpha(\vec{\theta}_L))\right]$, and has dimensions $M \times L$. The received signal vector $\vec{x}$ at the antenna array is obtained by superposition of signals due to all the paths, i.e.,

$$\vec{x} = A\vec{\Gamma} + \vec{n} \tag{11}$$

where $\vec{n}$ is the white Gaussian noise, $\Gamma = [\gamma_1' \quad \gamma_2' \cdots \gamma_L']$ is the vector of complex attenuations along L paths and A is the steering matrix. Since the steering vectors do not change across closely spaced subcarriers, we can formulate

$$\boldsymbol{X} = [\vec{x_1} \cdots \vec{x_K}] = \boldsymbol{A}\left[\vec{\Gamma_1} \cdots \vec{\Gamma_K}\right] + \boldsymbol{N} = \boldsymbol{A\Gamma} + \boldsymbol{N} \tag{12}$$

The basic idea of the MUSIC algorithm is eigenstructure analysis of an correlation matrix R_X of the CSI samples. Since the white noise is uncorrlated to the signal space, according to (9),

$$\begin{aligned} R_{\boldsymbol{X}} &= E[\boldsymbol{X}\boldsymbol{X}^H] \\ &= \boldsymbol{A}E[\boldsymbol{\Gamma}\boldsymbol{\Gamma}^H]\boldsymbol{A}^H + E[\boldsymbol{N}\boldsymbol{N}^H] \\ &= \boldsymbol{A}R_{\boldsymbol{\Gamma}}A^H + \sigma^2\boldsymbol{I} \end{aligned} \tag{13}$$

TABLE I
SUMMARY OF 1-D ESTIMATION OF MULTI-PATH FEATURE

	Snapshot Domin	CSI Shape
AoA	Transmitter, Frequency and Time	$(N, M \times K \times T)$
AoD	Recevier, Frequency and Time	$(M, N \times K \times T)$
ToF	Recevierm Transmitter and Time	$(K, M \times N \times T)$
Doppler	Recevierm Transmitter and Frequency	$(T, M \times N \times K)$

where $R_{\boldsymbol{\Gamma}}$ is the correlation matrix of the signal matrix, $\boldsymbol{I}$ is an identity matrix and σ^2 is the variance of noise. Since the signal and noise are independent of each other, the data covariance can be decomposed into a signal subspace and a noise subspace.

$$R_{\boldsymbol{X}} = U\Sigma U^H = U_S\Sigma_S U_N^H + U_N\Sigma_N U_N^H \tag{14}$$

where $U = [U_S \quad U_N]$ and $\Sigma = diag(\sigma_1^2 \cdots \sigma_M^2)$. U_S is a subspace formed by feature vectors corresponding to the largest $M - L$ eigenvalues, that is, a signal subspace, and U_N is a subspace formed by features vector corresponding to smallest L feature values, that is, a noise subspace. Since the signal subspace and noise subspace are orthogonal, we find that the steering vector $\alpha^{\vec{H}}(\theta)$ and the noise subspace of the signal subspace are also orthogonal, that is

$$\vec{\alpha(\theta)}^H U_N = 0, \theta = \{\theta_1, \cdots, \theta_n\} \tag{15}$$

In practice, each CSI sample requires multiple snapshots to average out the random noise, so we can take multiple snapshots for each CSI sample in its transmitter, frequency and time domian. We reshape the 3D CSI with shape (M, N, K, T) to $(N, M \times K \times T)$ to get $M \times K \times T$ snapshots for each CSI sample. And thus R_x can be estimated as,

$$\hat{R_{\boldsymbol{X}}} = \frac{1}{N \times K \times T} E[\boldsymbol{X}\boldsymbol{X}^H] \tag{16}$$

And since the orthogonality is not fully satisfied by actual estimation error and noise, the maximum optimized search of the spectrum function can be used to estimate the AoA:

$$\begin{aligned} \theta_{MUSIC} &= arg\max_\theta P(\theta) \\ &= arg\max_\theta \vec{\alpha(\theta)}^H \hat{U}_N \hat{U}_N^H \vec{\alpha(\theta)} \end{aligned} \tag{17}$$

The procedure to make estimation of ToFs, AoDs or Doppler vector is similar, expect we should reformat the steering vector and rearrange the CSI sample.

To estimate ToFs, for example, we can format the steering vector as $\alpha(\vec{\theta_l}) = \begin{bmatrix}1 & T(t_l) \cdots T(t_l)^{K-1}\end{bmatrix}$ and take multiple snapshots for each CSI sample in its transmitter, receiver and time domain. That is to reshape the 3D CSI with shape (M, N, K, T) to $(K, M \times N \times T)$ to get $M \times N \times T$ snapshots for each CSI sample. Note that we can also make joint-estimation of any combination of the features by reformatting the steering vector.

V. Signal Calibration

As mentioned before, CSI measurements are sensitive to the location and orientation of the antennas, which restricts the real-world application of environment-invariant WiFi sensing. So in this section, as shown in Fig 3, we propose a novel method to show how to calibrate the relative position of the transmitter and receiver as if AoA and AoD of the direct path between them is 0 and the length of the direct path is l.

Without loss of generality, we assume the transmitter is fixed and the receiver is placed at random location with random orientation with respect to the transmitter. Assume the AoD of direct path between transmitter and receiver is θ before calibration and the ToF of the direct path is t, then the length of the direct path will be tc.

Considering the fact that the power of the signal will be linearly attenuated during the propagation in the air, we can get the amplitude calibration factor for the signal

$$C_{amplitude} = \frac{l}{tc} \tag{18}$$

Considering the phase shift by AoDs and change of path length, the phase calibration factor for m^th receiver of the signal will be

$$C_{phase}(m) = e^{-j*2\pi*m*d_R*sin(\theta)*f/c} * e^{-j*2\pi*m*(l-tc)*f/c} \tag{19}$$

And thus the calibrated CSI will be

$$H'_{m,n,k} = C_{amplitude} * C_{phase}(m) * H_{m,n,k} \tag{20}$$

VI. Experiments

A. Effect of Phase Offsets Removal

As shown in Fig 4 and Fig 5, the modified CSI phase response, obtained by Eq 5 for each packet, does not change even if the STO changes, and hence is free from the variations of STO. So, the ToF parameters estimated across packets using modified CSI are free from variance of changing STO. Noted according to [4], the CSI phases of two consecutive samples will be same, which is also shown in Fig 5.

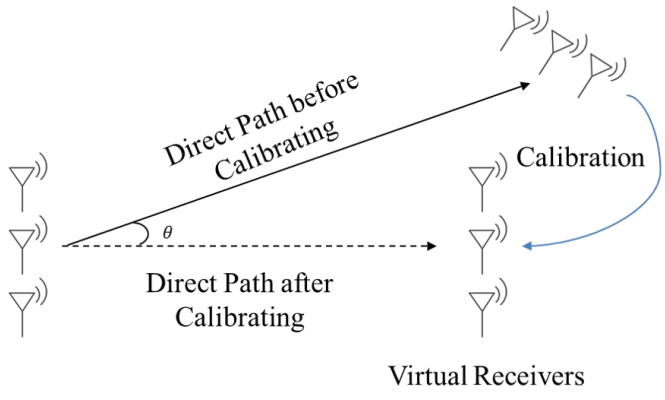


Fig. 3. The Target Effect of Calibration

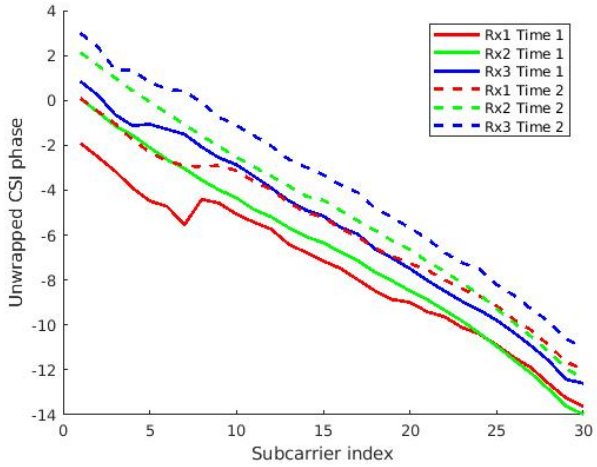


Fig. 4. Unwrapped CSI phase

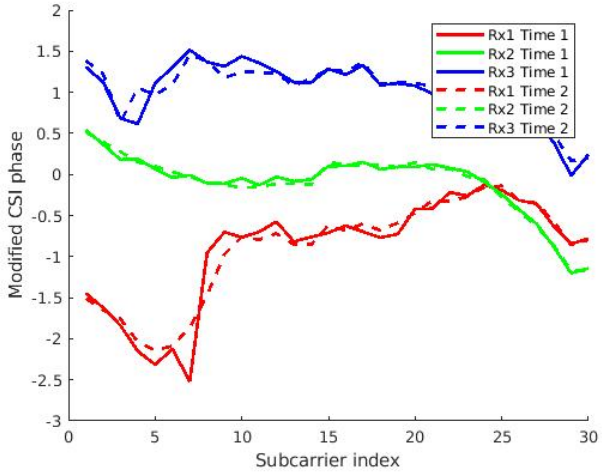
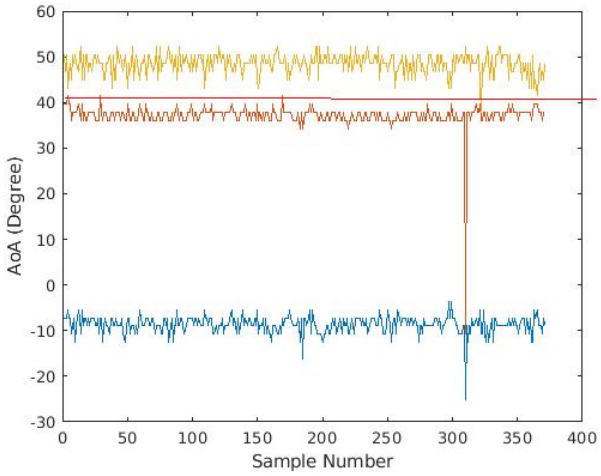


Fig. 5. Modified CSI phase

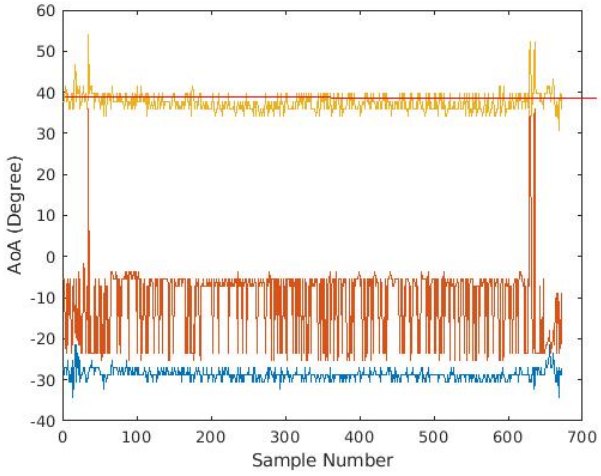
B. AoA Estimation Accuracy

For simplicity of demonstration, AoA is selected to be estimated using the aforementioned MUSIC algorithm. The original setting of WiFi antennas are shown in the Fig 1. To record CSI samples, we used a classic commercial WiFi Network Interface Card (NIC) and leveraged an open source tool [12], recorded CSI of 30 EM waves with a bandwidth of 20 MHz centering at the standard 5 GHz WiFi. The 5 GHz EM signal has a wavelength of around 2.6 cm. Similar to standard house-hold WiFi routers, we uniformly spaced three receiver antennas within a wavelength, 2.6 cm. This setting maximizes the difference of CSI captured at different receiver antennas.

To demonstrate the robustness of the algorithm, the receive antennas are set to make the direct path with AoA=40°, and the CSI is recorded under different static environments. As shown in Fig 6, the algorithm is capable to estimate the relatively accurate AoA even the environments vary. Also, the algorithm can also estimate the AoA for the main reflection paths of signal propagation.



(a)



(b)

Fig. 6. Estimated AoA (Direct Path = 40°) over Time with Different Static Environments

The estimation of AoA can aslo help to detect the human motion, since the propagation path of the signal will change when there is moving object in the environment. As shown in Fig 7, the estimation of AoA fluctuates when there is a human walking in the environment.

VII. Conclusion

In this work, we began to investigate the potential way to remove the random phase shift of Channel State Information (CSI) by estimating sampling time offset (STO), sampling frequency offset (SFO) and Cyclic Shift Diversity (CSD). Also, we discussed the methods to estimate to the import features of WiFi siganl propagation paths, including, angle of arrival, angle of departure, time of flight and Doppler velocity. In addition, we propose a novel way to calibrate the CSI data. Experiments show the Modified CSI phase is uncorrelated to the signal frequency and the estimation of AoA is not only

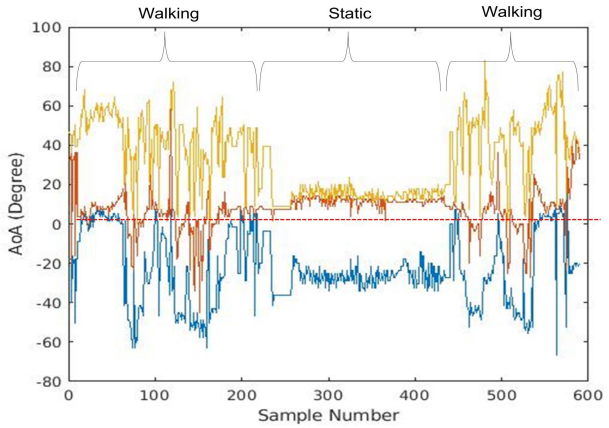


Fig. 7. Estimated AoA (Direct = 0°) over Time with Different Human Activity

robust to environment change but also help the detection of human movement.

Acknowledgment

I would like to thank Dr. Huang Dong and Mr. Fei Wang 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] F. Adib and D. Katabi, "See through walls with wifi!" in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 75–86. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486039

[2] M. Kotaru and S. Katti, "Position tracking for virtual reality using commodity wifi," *CoRR*, vol. abs/1703.03468, 2017. [Online]. Available: http://arxiv.org/abs/1703.03468

[3] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, 2013, pp. 71–84.

[4] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "Spotfi: Decimeter level localization using wifi," in *ACM SIGCOMM computer communication review*, vol. 45, no. 4. ACM, 2015, pp. 269–282.

[5] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang, "Wifinger: Talk to your smart devices with finger-grained gesture," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 250–261. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971738

[6] B. Fang, N. D. Lane, M. Zhang, A. Boran, and F. Kawsar, "Bodyscan: Enabling radio-based sensing on wearable devices for contactless activity and vital sign monitoring," 06 2016, pp. 97–110.

[7] B. Fang, N. D. Lane, M. Zhang, and F. Kawsar, "Headscan: A wearable system for radio-based sensing of head and mouth-related activities," in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April, 2016, pp. 1–12.

[8] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[9] Y. Ma, G. Zhou, and S. Wang, "Wifi sensing with channel state information: A survey," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 46:1–46:36, Jun. 2019. [Online]. Available: http://doi.acm.org/10.1145/3310194

[10] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 27–38. [Online]. Available: http://doi.acm.org/10.1145/2500423.2500436

[11] R. Schmidt, “Multiple emitter location and signal parameter estimation,” *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, March 1986.

[12] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, “Tool release: Gathering 802.11n traces with channel state information,” *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 53–53, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1925861.1925870

Using Fixed Route Transit to Improve Paratransit Service Quality

Rebecca Martin[1], Isaac Isukapati[2], Zachary Rubinstein[3], and Stephen Smith[4]

***Abstract*—Paratransit systems provide an equitable transit service to the elderly and handicapped through shared-ride, door-to-door transportation. The main objective of the dial-a-ride problem (DARP) is to optimally schedule and dispatch available vehicles to satisfy ride requests between given pick-up and drop-off locations at specified times. In practice, DARPs are often over-subscribed (i.e. demand exceeds capacity), resulting in increased wait times or failure to fulfill a subset of service requests within their constraints. Currently, the two ways of accommodating this extra demand that have been studied are relaxing the time constraints and increasing the fleet size. However, in contrast with the increased wait times and transportation cost of these two solutions, a rarely considered third solution is to use fixed-route transit to partially fulfill the trip. In this research, we are going to consider the static DARP scheduling problem with transfers between paratransit and fixed-route transportation. The crux of this idea is that the paratransit system services the last mile travel (pick-up & drop-off to/from the fixed-route transit locations), and the remaining part of the trip is serviced by the fixed-route transit. To evaluate the feasibility of the system, we analyzed the effect of this integration on overall service time. We found that using the bus system can greatly reduce the total travel duration for longer trips, but increases the ride time for shorter trips.**

***Index Terms*—Paratransit systems, Fixed Route Transit, Statistics, Simulation, Scheduling**

I. Introduction

Dial-a-ride paratransit systems play an instrumental role in providing equitable transportation services to special groups of the population, such as the elderly or handicapped. With a fare scheme comparable to that of a fixed-route transit, paratransit provides shared-ride, door-to-door service with flexible routes and schedules. The dial-a-ride problem (DARP) aims to optimally schedule and dispatch transit vehicles to satisfy requests for travel between pick-up and drop-off locations at specified times. A typical request in this context provides details on pick-up and drop-off locations, number of passengers, type of request, and optimal time windows within which the requests need to be fulfilled. The scheduling component can be static, dynamic, or a combination of both. In the context of static DARP, all the requests and available vehicle fleet are known well in advance. In a dynamic DARP, the requests are serviced on an ongoing basis with an ability to increase the fleet size as needed. In the hybrid approach, reservations made in advance permit the construction of day-ahead (off-line) schedules, while the same day requests and other events such as trip cancellations, vehicle breakdowns etc. can be re-integrated into the current schedule (on-line schedule).

In the real world, DARPs have limited resources, making them suffer from oversubscription, where there is more demand than capacity. This leads to increasing the wait time and could even result in failure to fulfill a subset of transit requests within the specified time windows. In principle, there are at least three possible ways to accommodate this extra demand: 1) constraint relaxation, 2) increase the paratransit fleet size, and 3) partial fulfillment of the trip via fixed-route transit. There is some literature on how to tackle the first two options, but there are still a number of issues that remain unresolved. Relaxing time constraints and thereby increasing passenger wait time leads to poor service quality and, in some cases, can compromise the safety of the passenger depending on where they are waiting. Increasing fleet size is not cost effective, especially as most of these programs are government subsidized.

The third option, though rarely considered, promises to solve the oversubscription problem while avoiding the problems presented by the first two solutions. The presence of ubiquitous connectivity and real-time communication presents opportunity to tackle the scheduling problem from a fresh perspective (paratransit $\rightarrow$ fixed-route transit $\rightarrow$ paratransit). In this research, we are going to consider the static DARP scheduling problem with transfers. The main idea here is that the paratransit transports the passenger from their pickup location to the nearest bus stop and then meets them at the bus stop closest to their destination to finish the trip, with the intermediate part of the trip being serviced by the fixed-route transit. Before we can evaluate the impact of the system on transportation costs and increased capacity, we must first ensure that this method avoids the problems presented by the other solutions. For this purpose, we will use total service time as the performance metric in an attempt to quantify quality of service.

II. System Architecture

A. Overall System Setup

In this system, we used historical paratransit data from Pittsburgh's ACCESS paratransit system and historical bus data from the P1 East Busway route of the Pittsburgh Allegheny County Port Authority bus system. For these experiments, we are only utilizing the P1 East Busway route, as the busway will be more likely to show improvements in ride time, due to the fact that other vehicles are not allowed to drive on the busway.

*This work was supported by the National Science Foundation

[1]R. Martin is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA rebecca_martin@asu.edu

[2]I. Isukapati is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA isaack@cs.cmu.edu

[3]Z. Rubinstein is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA zbr@cs.cmu.edu

[4]S. Smith is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA sfs@cs.cmu.edu

We started out with a set of 902 paratransit ride requests for one day. These requests were then augmented with boolean flags, marking whether or not that particular ride was a good candidate for partial fulfillment by the fixed-route transit system. A request was considered a good candidate if the specified pickup location was within 2 miles of a time point on the busway. A time point is a bus stop where the bus will wait at if it is running early so that it can get back on schedule. We chose to use the time points because they are usually main stops on the route and there is generally more time to transfer the passenger at these stops. The candidate requests were then split into two separate requests: one request took the passenger from the pickup location to the nearest bus stop and the other from a bus stop to their destination. Then, we used the given request data to generate the paratransit schedule.

The scheduler first used a greedy scheduling algorithm to construct an initial schedule. This initial schedule was then given as input to Generalized Task Swap [1], a local search algorithm, which optimized the schedule and made sure that all requests were scheduled within the given time constraints, if they had not managed to fit into the initial schedule. This scheduler was used to generate two different schedules. The first schedule used the original set of requests and had the paratransit system drop off the passenger at their specified destination. The second schedule, which used the augmented set of requests, was the hybrid schedule that incorporated the fixed route transit, as seen in Figure 2. If the request had been marked as a bus candidate, then the schedule had the paratransit system drop off the passenger at the busway stop nearest to their pickup location. Then, a new request was generated to transport the passenger from the bus stop nearest to their destination to the drop off location, shown in Figure 1. Else, if the request was not a bus candidate, the request was scheduled without modification and the passenger was dropped of at their specified destination.

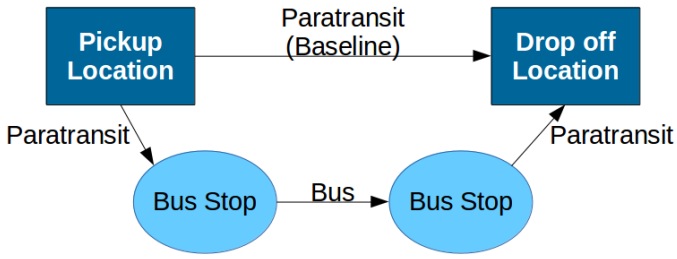


Fig. 1. First schedule represented by baseline route (above) and second schedule is represented by multi-legged route below

To test these schedules and incorporate some of the uncertainty present in carrying out the schedules, we built a statistical simulator in Python. The first schedule was tested in our simulator as a baseline with which we compared our results of the simulation of the second schedule. Both paratransit schedules were randomized by sampling from two uniform distributions. The first distribution determined whether the vehicle would arrive at its stop early, on-time, or late. If the vehicle was not on-time, the second distribution determined how early or late the vehicle was going to be, with a maximum of five minutes between the scheduled time and the randomized time. However, to preserve the spatio-temporal aspects of each trip, the randomized time between two stops was not allowed to be less than 90% of scheduled duration.

To create the bus schedules, we used the historical data to calculate the dwell time, the amount of time spent at each stop, and the link travel time, the time taken to drive between each pair of stops on the route, for each entry in the database. These values were then used to generate two Cumulative Density Functions, one for dwell times and one for link travel times. To randomize the schedule in a realistic manner, values were drawn from the appropriate CDF for each leg of the route and used as the duration for that part of the ride.

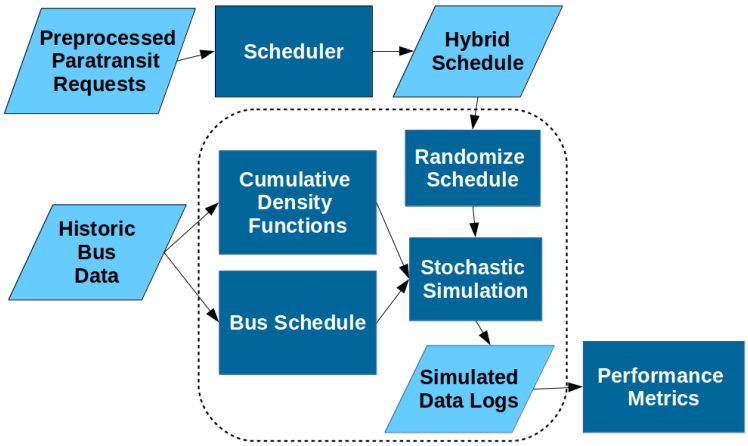


Fig. 2. Flowchart of the overall system, where the dashed box encapsulates the simulator pieces. The light blue parallelograms represent data that was either input or output for the system. The dark blue rectangles represent the different functions within the system that use the data along with their purpose.

B. Simulator Architecture

Within the simulator, we first divided the day into five windows of time, so that we could model different traffic conditions based on time of day. This allowed our model to more closely follow the actual bus schedule. For the transit vehicles, we modeled each Bus and Paratransit Vehicle separately. The Buses were generated and added to the fleet on an as-needed basis, while the Paratransit Vehicles were all generated and added to the fleet at the beginning of the day. Each Bus had its own Fixed Route Transit Schedule to keep track of the times at which it would arrive at and depart from the stops on the route. The Fixed Route Transit Schedule also contained the dwell time and link travel time CDFs, which were generated using only the historical data from within the same time window during which the bus started its route. Inside the simulator, there was only one global Paratransit Schedule, and each Paratransit Vehicle fetched its individual schedule directly from there at the beginning of the day.

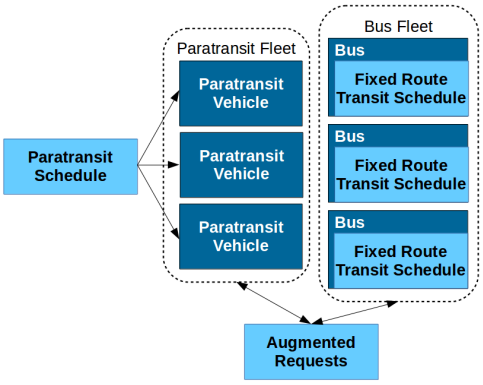


Fig. 3. Simulator structure

III. Results

The main focus of our experiments was to test the effect of the bus headway on overall service time. Bus headway is the time between consecutive buses at any given stop. We ran the simulation fifteen times for each condition and computed the time between initial pickup and final dropoff as the total trip duration. Our control experiments were the simulations of the first schedule that did not incorporate the bus system at all.

For the experimental simulations, we tested the second schedule with headways of five minutes, ten minutes, fifteen minutes, and thirty minutes. As our performance metrics, we found the minimum, maximum, average, and median of the total trip durations. This analysis was completed for the entire set of requests and for the subset of candidate requests.

Table 1 shows the duration statistics for all of the requests for the entire day, while Table 2 shows the duration statistics for the subset of candidate requests. As the candidate subset only accounts for 14% of the rides, the degree to which the fixed route transit affects the trip durations is not initially apparent. In Table 1, both the minimum and the maximum duration experience a negligible change as the bus headway increases. Likewise, while the average and median do increase with the headway, it is not a significant effect.

However, when the candidate subset is analyzed on its own, the result of adding the bus becomes much more apparent. While the minimum durations for the experimental cases are longer than for the baseline simulations, the maximum durations for all of the tested headways are significantly shorter. In addition, between the baseline simulations and the 5 min headway tests, the medians are four minutes apart, while the averages are the exact same. This suggests that the travel time of the shorter candidate trips increased and that of the longer candidate trips decreased.

Table 1: Duration statistics from entire set of requests

Headway	Min	Max	Average	Median
Baseline	00:00:56	01:37:22	00:17:49	00:13:31
5 min	00:00:57	01:36:47	00:18:23	00:14:40
10 min	00:00:57	01:36:46	00:18:45	00:15:04
15 min	00:00:59	01:36:59	00:19:04	00:15:22
30 min	00:00:57	01:37:15	00:19:57	00:15:38

Table 2: Duration statistics from candidate subset of requests

Headway	Min	Max	Average	Median
Baseline	00:02:48	01:17:31	00:21:00	00:16:26
5 min	00:05:40	00:54:37	00:21:10	00:20:02
10 min	00:06:04	00:58:27	00:23:56	00:22:40
15 min	00:06:20	00:59:36	00:25:58	00:24:11
30 min	00:06:44	01:13:10	00:32:24	00:30:32

In order to visualize the effect of utilizing fixed route transit, we plotted the Cumulative Density Functions (CDFs) of the overall service time distributions, measured in seconds. Figure 4 shows the CDFs of the overall travel time for each headway, plotted with the CDF of the baseline distribution for reference. For the simulations with large headways, the added serice time to the subset of requests is significant enough to show a visible difference in the overall time distribution for all of the trip requests. However, on an overall scale, the experiments with a shorter headway give an overall service time distribution that is almost the same as the baseline simulations.

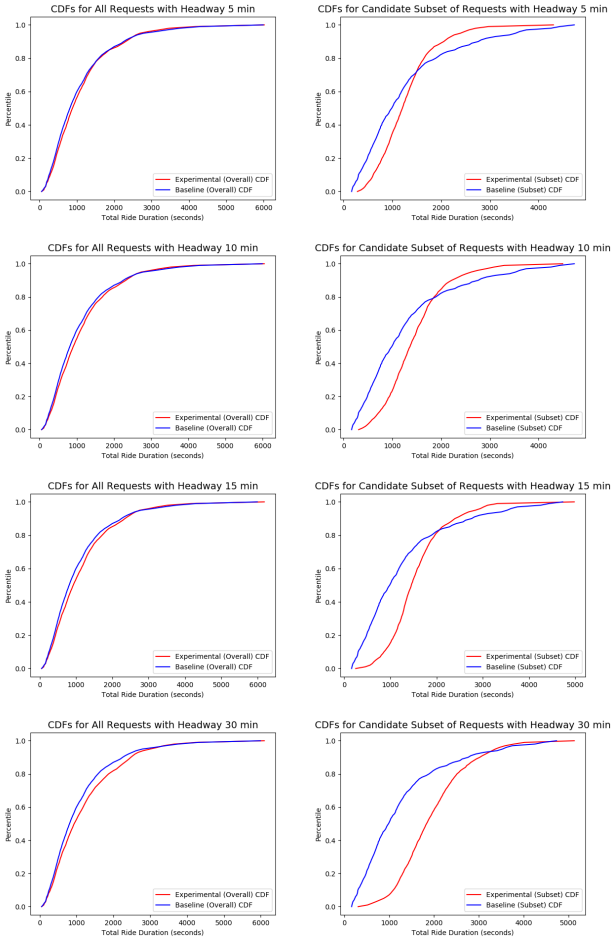


Fig. 4. Overall travel time distributions for all requests (left column) vs subset of candidate requests (right column). The distribution for each headway has been plotted along with the baseline distributions.

Figure 5 shows the CDF of the overall travel time for the subset of candidate requests for both the baseline simulations

and the experiments with a headway of fifteen minutes. This confirms that adding the fixed route transit mainly benefits the rides that have a duration above a certain threshold, which varies based on the headway value. The benefit of traveling on the busway is not significant enough for the shorter rides to outweigh the extra distance traveled.

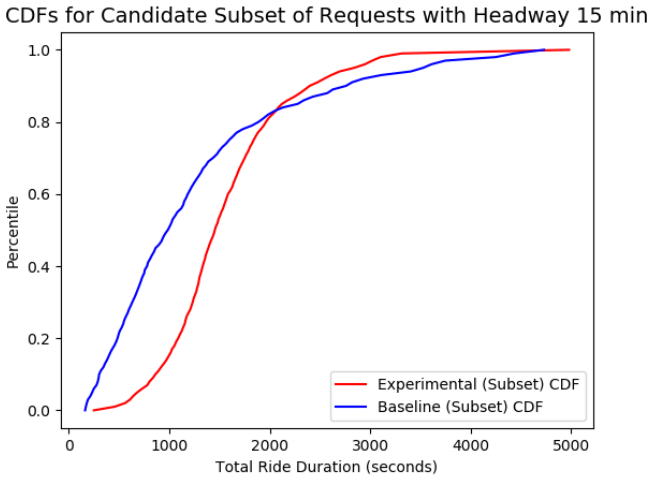


Fig. 5. The CDF of the overall travel time distribution of the subset of candidate requests when the headway was set to 15 minutes.

IV. Conclusion

In this paper, we explore the feasibility of integrating fixed route transit into paratransit schedules in order to reduce overall service time. We chose which rides to use the bus with based on the proximity of the pickup location to a bus stop and generated a possible schedule to fulfill these requests. The baseline schedule and experimental schedule were both tested in a statistical simulator written in Python. We found that adding fixed route transit decreases ride duration for long trips, but increases duration for short trips.

V. Future Work

Based on our results, we are going to add minimum distance covered on bus as candidacy constraint, that way we avoid the extra time added to the shorter ride requests. Now that we have shown that this solution can be implemented such that minimal extra time is added to some requests, which was an issue with the other solutions proposed to avoid oversubscription, we will analyze the system using other performance metrics such as cost and capacity. To more realistically simulate traffic conditions, we will increase standard deviation of the paratransit travel time distributions and test the simulations using dynamic paratransit scheduling. On the scheduling side, we will try scheduling the paratransit rides to a particular bus time, instead of waiting for next bus. With this, we will be able to test the feasibility of holding the bus based on scheduled paratransit arrival at stop.

Acknowledgment

This work was supported by the National Science Foundation Robotics Institute REU. Rebecca would like to thank Dr. Isaac Isukapati for his guidance and mentoring. She would also like to thank Dr. Zachary Rubinstein and Dr. Stephen Smith for their advice and feedback. Finally, she would like to thank Rachel Burcin and Dr. John M. Dolan for their tireless commitment to the Robotics Institute Summer Scholars program.

References

[1] Z. B. Rubinstein, S. F. Smith, and L. Barbulescu, "Incremental management of oversubscribed vehicle schedules in dynamic dial-a-ride problems," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[2] Z. B. Rubinstein and S. F. Smith, "Paratransit scheduling and routing," in *Accessible Public Transportation*. Routledge, 2017, pp. 83–89.

[3] ——, "Dynamic management of paratransit vehicle schedules," in *Proceedings of the International Workshop on Scheduling and Planning Applications*, vol. 124, 2011.

[4] M. Aldaihani and M. M. Dessouky, "Hybrid scheduling methods for paratransit operations," *Computers & Industrial Engineering*, vol. 45, no. 1, pp. 75–96, 2003.

[5] L. Fu, "A simulation model for evaluating advanced dial-a-ride paratransit systems," *Transportation Research Part A: Policy and Practice*, vol. 36, no. 4, pp. 291–307, 2002.

[6] R. Lave and R. Mathias, "State of the art of paratransit," *Transportation in the New Millennium*, vol. 478, pp. 1–7, 2000.

[7] J. P. Franklin and D. A. Niemeier, "Discrete choice elasticities for elderly and disabled travelers between fixed-route transit and paratransit," *Transportation research record*, vol. 1623, no. 1, pp. 31–36, 1998.

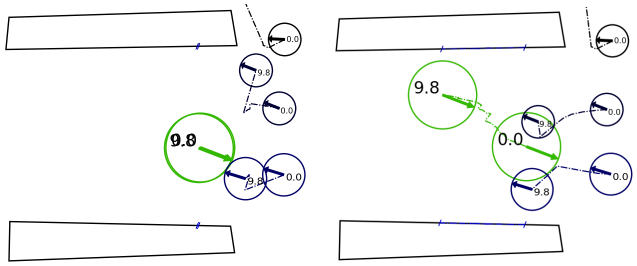
Learning to Fail: Failure Plans and Predictions for Crowd Navigation

Patrick Naughton[1], Tushar Kusnur[2], Ishani Chatterjee[3], and Maxim Likhachev[4]

***Abstract*— Probabilistic planners have demonstrated promising results in dealing with robotic navigation under uncertainty [1], for example, when parts of a map are unknown or the robot must navigate amongst other unpredictable agents. To compute trajectories, probabilistic planners rely on atomic actions which are strung together to form a path. Planners will often invoke a lower level controller to execute these actions. Existing work focuses on situations in which the humans in the scene move such that completion of the desired action is always feasible. Little attention has been paid to scenarios in which humans make reaching the goal waypoint impossible or prohibitively costly. Current methods may simply attempt to replan in these cases which can be too slow to avoid imminent collisions. Additionally, if the waypoint is unreachable or costly to reach, this approach may incur a higher cost than abandoning the waypoint and moving to a different location first before replanning. This paper presents a reinforcement learning based approach to developing *failure controllers* for exactly these situations which direct the robot to an optimal failure location if its original goal waypoint becomes unreachable. We also present a *predictor* which forecasts the outcome of each step of the controller at planning time. This controller and predictor make the long-term planner more robust by giving it a notion ahead of time of the potential results of executing a given controller and help the robot navigate more safely and efficiently when actually executing its trajectory. We compare the trajectories produced by our controller to trajectories produced by a social forces [2] controller and by the robot remaining still to show how our failure controller balances safety with time and distance optimality.**

I. Introduction

Pedestrian crowds present a challenging navigation environment for a mobile robot. The robot must comply with latent social rules governing its trajectory while simultaneously reaching its goal in a reasonable amount of time. As robots move into closer contact with humans, advanced methods for navigation in crowds will gain increased importance. Probabilistic planners, which make navigation plans in the belief space, show potential to make headway in this problem and have demonstrated success dealing with uncertainty like that experienced navigating in a crowd. To form their plans, these planners rely on smaller atomic actions which are strung together to form an overall trajectory. Some of these actions are learned by presenting the robot with its starting pose and a waypoint pose and running simulations with pedestrians or other agents [3]. In general, these controllers guiding each action deal with unexpected situations in which their original trajectory to their waypoint becomes infeasible by simply replanning or stopping altogether. Replanning in some cases may take too long to avoid an imminent collision. Additionally, when the waypoint becomes completely unreachable, replanning or stopping may result in a more costly trajectory than simply abandoning it and seeking a safe location from which the higher level planner can generate a new trajectory to the overall goal. We address this issue by developing a failure controller aimed explicitly at motion control for the robot when it has no goal.



(a) (b)

Fig. 1: The green circle represents the robot, the blue circles represent different humans and the black polygons represent the walls of the corridor. The arrows indicate the heading of each agent. Only the initial and final locations of each object are shown. The numbers indicate the time step at which the snapshot was taken. This is an example of a situation in which the robot stopping (a) will cause more collisions and intrusions than following a failure controller which in this case simply moves the robot along the corridor out of the way of humans (b).

Developing such a controller is useful because it gives long term planners a contingency plan and predicted future state at planning time so they can more efficiently explore the search space. The failure controller itself also helps ensure that the robot obeys a safe policy even when it has no waypoint to navigate to. For example, in some situations like that in Figure 1a, staying still will cause the robot to be a hindrance to the humans and it will likely intrude into their personal space. Moving away to a different area of the scene would give the robot a safer place to replan from.

This work facilitates combining learning with planning for long term crowd navigation by providing a reinforcement learning based failure controller and a supervised learning based predictor for forecasting the results of the controller. Specifically, this paper assumes that an existing probabilistic planner uses a set of motion primitives and controllers to create navigation plans. The set of controllers includes some

[1]Patrick Naughton is a junior studying Electrical Engineering and Computer Science at Washington University in St. Louis, St. Louis, MO 63130, USA. patrickrnaughton@wustl.edu

[2,3,4]Tushar Kusnur, Ishani Chatterjee, and Maxim Likhachev are with Search Based Planning Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {tkusnur, ichatter, mlikhach}@andrew.cmu.edu

to execute complex behaviors that interact with pedestrians. Each of these controllers has some probability of success which is associated with one goal and trajectory, and some probability of failure which is associated with a different trajectory. Success cannot be guaranteed for the controllers because the surrounding humans may interact with the robot or each other in unexpected ways. Note that the failure controller's objective differs from that of a traditional navigation planner in that the failure controller is not given an explicit goal; rather, it attempts to extract the optimal goal and trajectory from its environment.

The main contributions of this work are:

1) A reinforcement learning based controller for commanding robot trajectories in cases of failure.
2) A predictor that forecasts a distribution of locations the robot will end up in at each time step.

The design philosophy taken by this approach separates controllers for individual actions from the long term planner. This means the results presented here could easily be adapted to work with any planner.

The remainder of this paper is organized as follows. Section II discusses background information on the problem presented here including related work and a formalized statement of the problem. Section III lays out the approach taken to construct and train both the failure controller and the predictor. Section IV gives the results of simulation experiments and compares the method presented here with other methods. Finally, Section V draws conclusions from these experiments and suggests future research directions.

The code of our approach is available here: `https://github.com/patricknaughton01/LearnController`.

II. Background

A. Related Work

Several methods use explicit models of human behavior to achieve smooth, predictable robot navigation among pedestrians. Trautman et al. model the interaction between the robot and pedestrians as an extension of an interactive Gaussian process that accommodates multiple goals [4]. The social forces model [2] treats humans and the robot in question as masses subject to Newtonian dynamics and applies fictitious forces to them to predict and plan trajectories. It recomputes these forces and their effects on robot motion at each time step to determine how the robot should move. These techniques however rely on hand-crafted models of human behavior to achieve their results and handle unexpected or uncooperative human actions by simply replanning using the same model. The social forces model in particular does not demonstrate robust navigation plans and will sometimes exhibit oscillatory behavior in more crowded or narrow areas [2].

Another approach uses inverse reinforcement learning to learn latent, possibly stochastic social rules humans observe when navigating in crowds [5]. This method uses example trajectories recorded from humans or gathered from teleoperated runs. This approach however is extremely unlikely to observe failed trajectories where a human attempts to execute some navigation plan and is forced to completely abort their initial goal. If a human attempts to overtake someone else, for example, they have many contingency options in the case where the other person is either intentionally or unintentionally uncooperative. For example, they could use verbal communication or body language to more explicitly communicate their intentions, options which are not available to many mobile robots. For this reason, inverse reinforcement learning will likely be unable to formulate a useful model for navigation when situations such as these occur.

Advances in deep reinforcement learning have led many to apply it to this problem. After demonstrating its capability to match and in some cases exceed human performance in a variety of video games [6], much research has focused on applying it to different domains. It has successfully been applied to the social robot navigation problem using a variety of different models [7], [3]. Reinforcement learning is particularly suited to this application as noted in [7] because it is extremely difficult to specify what the optimal action for a robot to take is but it is much easier to alert the robot when it takes a socially unacceptable or unsafe action. Previous work has focused on using reinforcement learning to develop policies that generate optimal (in terms of time) paths to a robot's goal in the presence of humans or other autonomous agents. These policies however generally assume the goal is reachable and do not make contingency plans if that assumption turns out to be incorrect. Additionally, the agent is explicitly given a goal to reach by the experimenters; we wish to navigate in the case of failure at which point there is no obvious goal.

The above methods all either deal with failure at execution time by simply replanning or do not consider failure to reach the goal at all. We depart from this paradigm by designing a controller specifically targeted at producing trajectories when the robot's original goal is no longer reachable.

B. Problem Statement

Consider a robot using a probabilistic planner to navigate in a pedestrian environment. The environment contains both static and dynamic obstacles in the form of humans and other objects. This robot has a set M of motion primitives which it can use to perform simple movements. The robot also has a set C of controllers which it can execute to perform more complex behaviors that may require cooperation from humans. Because these controllers have some probability of failing, the robot needs a contingency plan to follow that will return it to a safe location. Moreover, we would like to know at planning time how this trajectory is likely to look so that the high-level probabilistic planner can account for this. Thus, a predictor that can determine the distribution of the next locations the robot will occupy at each time step is sought.

This paper demonstrates our framework by applying it to a *barge-in* controller. This controller addresses a situation in which humans block a narrow corridor and the robot wishes to move past them. Ideally, the robot would drive

towards the group of people who would then part enough for the robot to reach its goal. However, if the people do not move—for example, if they do not notice the robot or they cannot infer its intention to move past them—the robot needs a contingency plan that results in safe behavior and brings the robot back to a location from which it can replan.

We model the problem of finding the optimal failure policy as solving a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$ with an infinite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, state transition matrix $\mathcal{P}(s_t, a_t, s_{t+1})$, reward function $\mathcal{R}(s_t, a_t, s_{t+1})$ and discount factor $\gamma \in (0, 1)$ [8]. The robot seeks an optimal policy $\pi^*(a_t \mid s_t) = P(a_t \mid s_t)$ where optimal indicates that the policy maximizes the overall expected discounted reward the robot receives (referred to as the "return" of the policy) from the current time step t onwards:

$$\sum_{k=0}^{\infty} E[\gamma^k \mathcal{R}(s_{t+k}, a_{t+k}, s_{t+k+1})] \tag{1}$$

To aid in finding this policy, we define a $Q_\pi(a_t, s_t)$ function which estimates the return of executing action a_t from state s_t and following policy π from that point onwards:

$$Q_\pi(a_t, s_t) = \sum_{k=0}^{\infty} E[\gamma^k \mathcal{R}(s_{t+k}, \pi(a_{t+k} \mid s_{t+k}), s_{t+k+1})] \tag{2}$$

$$\begin{aligned} Q_\pi(a_t, s_t) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s_t, a_t, s')[\mathcal{R}(s_t, a_t, s') \\ + \gamma \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q_\pi(a', s')] \end{aligned} \tag{3}$$

Assuming π is greedy, we can rewrite[1]

$$Q_\pi(a_t, s_t) = \mathcal{R}(s_t, a_t, s_{t+1}) + \gamma max_a Q_\pi(a, s_{t+1}) \tag{4}$$

We denote the Q function of the optimal policy as Q^*. Q^* gives a convenient way to determine the optimal action for any state given that the action space is finite and small because we can simply find the action with the largest Q^* value and execute it.

Finally, we wish to create a predictor which estimates a distribution of next locations for the robot given the initial state. This predictor takes in the robot's initial state and assumes its distribution of subsequent locations is Gaussian. It learns to output the parameters of this two dimensional distribution that maximize the likelihood of the observed next positions.

III. Approach

A. Failure Controller

1) State Representation: Because the MDP we have described has infinitely many states, it is impossible to store a Q value for each state-action pair explicitly. We base our state characterization on that presented in [3] with slight modifications because of our controller's lack of a goal. An initial matrix is constructed with the same number of rows as there are agents and obstacles in the scene (including the robot). Each row of the matrix takes the form $(s_{\text{pref}}, h, r, v_x, v_y, p'_x, p'_y, v'_x, v'_y, r', d, r_{\text{sum}})$ where

- s_{pref} is the preferred speed of the robot.
- h is the heading of the robot w.r.t. the global coordinate frame.
- r is the radius of the robot.
- v_x and v_y are the x and y velocities of the robot w.r.t. the robot's coordinate frame.
- p'_x and p'_y are the x and y coordinates of the other agent w.r.t. the robot's coordinate frame.
- v'_x and v'_y are the x and y velocities of the other agent w.r.t. the robot's coordinate frame (note, this is not relative to the robot's velocity, just relative to its rotation).
- r' is the radius of the other agent.
- d is the Euclidean distance between the robot and the other agent.
- r_{sum} is the sum of the radii of the robot and the other agent.

The first five components of each row are referred to as the robot's `self_state` because they pertain specifically to the robot.

This is augmented by a series of occupancy maps, one for each human and obstacle in the scene and one for the robot itself. Each occupancy map is centered on and aligned with its respective agent and their heading and is discretized into a number of squares. Each square indicates in a binary fashion whether or not it contains an object. Each square also contains the average velocity of all the objects within that square (simply set to 0 if the square is unoccupied). In our case, each occupancy map is discretized into 16 (4x4) 1 unit squares. This collection of occupancy maps and information about the robot's dynamics form the robot's *state*. Note that the dimensions of the state matrix will vary depending on the number of humans and obstacles in the scene.

For the purposes of computing the position, velocity, and heading of a given obstacle we simply consider the point on the obstacle which is closest to the robot. Any obstacle will always have a velocity of 0 and a heading along the robot's x-axis.

2) Q Function: There are 35 different available actions in any given state. One is to do nothing (remain in the same location). There are then 16 directions the robot can move w.r.t. its heading which are evenly spaced from 0 to 2π (we assume a holonomic robot) and the robot can travel at either half or full speed in each of these directions. Finally, the robot can also rotate in either direction by a small amount. We represent Q_π using a deep neural network whose architecture is based on that presented in [3] which showed that it could generate satisfactory goal-oriented trajectories navigating amongst humans. The state of the robot is first given to a Multi-Layer Perceptron (MLP). This output is processed by two additional MLPs, one of which computes weights and one of which computes features. The weight MLP also receives the average of the input MLP's outputs across the different rows of the state matrix. The weights

[1]This rewrite also makes use of the fact that the ground-truth next state is an unbiased estimator of the expected next state.

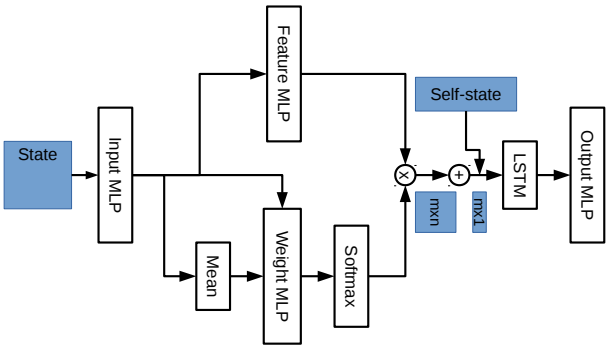


Fig. 2: Architecture of our network for learning the Q_π function. Since the number of humans in the scene could vary, the state input is not a constant size. The weighted features get summed across each agent to form a constant size vector before being given to the LSTM.

returned from this MLP are then passed through a softmax layer. The dot product of the features and softmaxed weights is then computed which yields a constant size vector (independent of the number of agents and obstacles in the scene). This result, along with the robot's `self_state` are then fed into a layer of LSTM cells. The outputs of these cells are processed by a final MLP which generates Q values for each of the 35 different possible actions simultaneously.

The different components of the overall network have the following dimensions: [60, 80], [80, 120], [160, 64, 31, 1], [86, 256], [256, 128, 64, 35] for the input MLP, feature MLP, weight MLP, LSTM, and output MLP respectively.

3) Reward function: The reward function $\mathcal{R}(s_t, a_t)$ at each time step is computed in three parts: $\mathcal{R}_{collision}$, $\mathcal{R}_{movement}$, and $\mathcal{R}_{smoothness}$ for collision, movement, and smoothness rewards respectively. The components are defined in the following way:

$$\mathcal{R}_{collision} = \begin{cases} -1 & d < 0 \\ -0.25 & 0 \leq d < 0.2 \\ 0 & else \end{cases}$$

Here, d denotes the distance between the robot and an obstacle or another agent. Note that this reward is applied individually with respect to each human and obstacle in the scene so if the robot is colliding with two humans simultaneously it will receive a reward of -2. When the robot occupies the second region ($0 \leq d < 0.2$) we refer to this as an *intrusion* and call the upper bound of 0.2 the *intrusion threshold*. This component disincentivizes colliding with humans and obstacles. It is based on the reward function used in [3]. Additionally, for each human or obstacle the robot is forecasted to collide with or intrude into (based on current velocities) the robot receives a reward of -0.05 (for predicted collisions) and -0.0125 (for predicted intrusions). This was done to make the reward function smoother.

$$\mathcal{R}_{movement} = \begin{cases} 0 & a_t = 0 \quad (remain\ still) \\ -0.01 & else \end{cases}$$

This component is meant to very slightly punish moving so that areas farther from the robot's initial location are less desirable. It was added because, all other things equal, areas closer to the robot's initial position are preferred because they can be reached more quickly.

$$\mathcal{R}_{smoothness} = \begin{cases} 0 & a_t = a_{t-1} \vee a_{t-1} = 0 \\ -0.01 & else \end{cases}$$

The smoothness component of the reward incentivizes using the same action repeatedly so that the robot prefers smooth trajectories. These trajectories are qualitatively more predictable for humans which makes them easier to navigate around and result in plans which are generally easier to execute.

The total reward at each time step is simply the sum of these three components.

4) Training: To simulate the robot moving about in a pedestrian environment the RVO2 library as well as its Python wrapper were used [9], [10]. This library provides a convenient way to simulate an environment with multiple autonomous agents and obstacles. It uses the Optimal Reciprocal Collision Avoidance (ORCA) [11] algorithm to generate safe velocity vectors for each agent to follow to reach a predefined goal.

For the barge-in scenario, each training episode starts with the robot right behind a group of humans blocking a narrow passage as though it had approached attempting to move past them. The width of the corridor, number and size of the humans, and the initial position and size of the robot are all initialized with small random perturbations. Each human sets a goal at a random location inside the corridor which would force the robot to abort its normal plan. The robot then begins executing actions according to an epsilon-greedy strategy, choosing a random action with probability $\epsilon \in [0, 1]$ and otherwise choosing the action with maximum estimated Q value. When the robot selects an action, this only sets its preferred velocity. The underlying simulator may modify the actual velocity it commands if an imminent collision is detected.

We use the deep Q-learning algorithm with experience replay to train our Q-network [12]. Two copies of the Q-network are stored, one called the `policy_model` and one called the `target_model`. The simulated robot follows an epsilon-greedy policy and records a tuple containing the state, action, reward received, and next state at each time step in a memory buffer. It then samples a batch of these tuples and updates the `policy_model`. Pseudocode for this algorithm is presented in Algorithm 1.

The `target_model` and `policy_model` were implemented using PyTorch. An Adam optimizer with a learning rate of 0.001 was used to make updates to the

Algorithm 1: Deep Q-Network with Experience Replay and Target Network

```
Initialize replay buffer R_B
Initialize policy_model network N_P
Initialize target_model network N_T ← N_P
Initialize cumulative time steps t_c to 0
for episode = 1, E do
    for t = 1, T do
        Increment t_c
        With probability ϵ select an action a_t randomly
        Otherwise, select a_t = argmax_a N_P(a, s_t)
        Record the tuple ⟨s_t, a_t, r_t, s'_t⟩ in R_B
        Sample a batch of tuples from R_B
        Set y_i = r_i + γ max_a N_T(a, s'_i)        ▷ Eq. 4
        Accumulate loss between y_i and N_P(a_i, s_i)
        Perform a gradient descent step on N_P
        if t_c mod t_update = 0 then
            Update N_T ← N_P
        end
    end
end
```

`policy_model`'s weights and each MLP had a dropout probability of 0.1. ϵ started at 1.0 and linearly annealed to 0.1 over 250000 time steps. γ was set to 0.9. Each episode consisted of 100 time steps (corresponding to approximately 10 seconds). The mean squared error loss function was used to compute the loss between the `policy_model`'s predictions and their targets.

B. Prediction

We assumed that the robot's next location at each time step would form a Gaussian distribution. After learning an optimal Q function, sample trajectories were generated using a greedy policy. At each time step, the robot's initial state and next location were recorded. These examples were used to train a `predictor` network which has the same architecture as the `policy_model` and `target_model` but only five outputs. These five outputs correspond to μ_x, μ_y, $ln(\sigma_x)$, $ln(\sigma_y)$, and $tanh^{-1}(\rho)$ which characterize a two-dimensional Gaussian distribution of possible locations the robot could end up in. The network was then trained on the sample trajectories using a negative log-likelihood loss to find the maximum likelihood estimates for the parameters conditioned on the initial state of the scene.

IV. Results

A. Simulation

To evaluate the final network, variants of its training environment were generated by randomly perturbing the locations and shape of the corridor, the number of humans in the scene, the starting locations and radii of the robot and humans, and the goals the humans try to reach. Trajectories such as the one shown in Figure 3 were first qualitatively evaluated for reasonableness. In general, we would like the robot to move smoothly away from humans coming towards it while simultaneously stopping as soon as it safely can. These criteria are quantified by the metrics we used to evaluate each path.

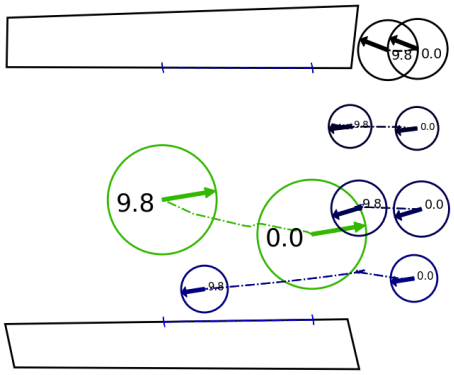


Fig. 3: Example trajectory generated by our failure controller.

B. Metrics

Other methods of social navigation were implemented for comparison with the result presented in this paper. To the authors' knowledge, the vast majority of existing controllers for navigation require that some external agent (either a planner or a human operator) provide a goal. Because of this, our selection of competing methods was rather limited. It was decided to compare to a social forces controller and a controller which simply has the robot hold its position when it fails. These methods were chosen because the social forces model has demonstrated success in the past for some social navigation problems [13] and provides a simple means of operating without an explicit goal (i.e., simply do not compute a force directed towards the goal). Doing nothing was chosen because it is an intuitively appealing heuristic solution to this problem and is often times used when traditional planners are replanning.

All three controllers were run in 100 randomly generated variants of the barge in scenario for 10 seconds before being cut off. Their trajectories were then evaluated on: length, time elapsed, angular distance, number of collisions, and number of intrusions. The time elapsed for a trajectory was computed as the time between the start of the episode and when the robot makes its final move (which could happen before 10 seconds). Angular distance was computed as the sum of differences in headings between consecutive time steps and is used to give a sense of path smoothness. The number of collisions was calculated by accumulating the number of obstacles and agents the robot was in collision with in each time step. Similarly, for each time step, the number of agents and obstacles for which the robot was within the intrusion threshold was counted to find the number of intrusions a trajectory incurred. These metrics are based on those used in [14] to evaluate robots designed to navigate in human crowds. Table I summarizes these results and Figure 4 provides examples of the observed trajectories.

In general, the social forces model did a very good job

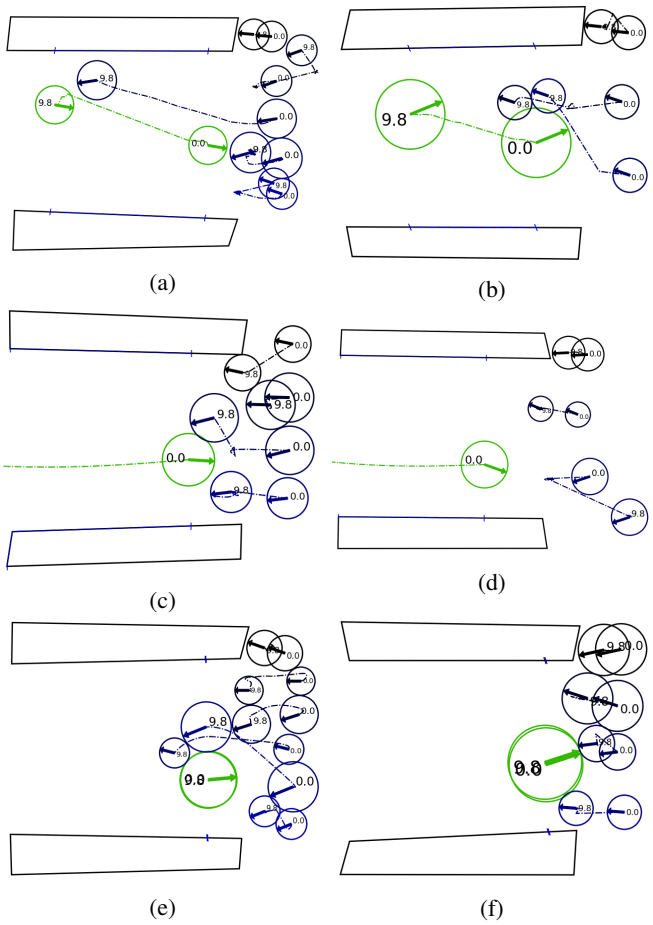


Fig. 4: Sample trajectories for: our controller (a, b), a social forces controller (c, d), and a do nothing controller (e, f). The initial and final locations of the robot and all the humans are shown except for the social forces trajectories because their final locations are so far away. They continue in an approximately straight line away from the humans until they reach their time-out.

TABLE I: Comparison between the presented controller, a social forces controller, and a do nothing controller. The mean of each metric over 100 runs is presented with the standard deviation in parentheses. Note that although the "do nothing" controller attempts to hold its position the underlying simulator will sometimes force it to move if an agent moves towards it so that its distance travelled is not always 0.

	Mean (Standard Deviation)		
	RL (Ours)	Social Forces	Do Nothing
Path Length	1.23 (0.74)	14.70 (0.00)	0.07 (0.12)
Angular Distance	21.15(13.27)	4.18 (1.47)	10.28 (6.95)
Time	5.61 (2.96)	10.00 (0.00)	3.39 (3.45)
Collisions	0.02 (0.14)	0.00 (0.00)	0.75 (4.74)
Intrusions	34.72(47.11)	0.76 (2.41)	190.58(60.71)

of avoiding people and obstacles but has no real notion of when to stop. This behavior is expected because without a goal attracting it the controller simply attempts to move as far away from everything as possible as quickly as possible. This causes it to use all of its available time and travel a rather excessive distance. On the opposite end of the spectrum, having the robot hold its position unsurprisingly causes it to travel very little distance and finish its trajectory quite quickly, but also generates many collisions and intrusions.

C. Predictor

For the predictor, we did not benchmark it against anything because we simply wished to show that it produces reasonable estimates with relative uncertainties that correspond to how chaotic the environment is. Namely, we wish to show that the uncertainty in situations that are relatively stable and predictable is lower than in situations with a high amount of movement, and that the predicted distribution tends to reflect the actual next location of the robot. As can be seen in Figure 5, evaluations at time steps in which the robot moves very little or not at all tend to have lower uncertainty and higher accuracy than evaluations when the robot makes large motions. This is expected and reflects the fact that the predictor is less confident when the robot moves because it has much more potential variability than when remaining in place. Note additionally that the axis along which the predictor is most uncertain tends to align with the direction the robot is most likely to move. This is most apparent in Figure 5d where the robot is more likely to move horizontally rather than vertically to distance itself from the people.

V. Conclusion

This paper uses a reinforcement learning based approach to develop a failure controller which guides a robot along a trajectory in cases when its original waypoint becomes unreachable. We also develop a predictor to forecast the distribution of next locations the robot will occupy at any given time step at planning time. This failure controller effectively balances the extremely cautious behavior of moving as far away from humans and obstacles as possible with the time and distance optimality of simply doing nothing. The specific point at which our controller lies along this spectrum is dictated by the relative magnitudes of the penalties specified in the reward function. The predictor generates satisfactory forecasts with uncertainties that tend to reflect the variability of the robot's options at any given time step.

An clear future direction for this research is to test this method on other controllers to, for example, overtake or cross trajectories with a person. Another useful extension would address the issue of when, during execution time, the robot should switch from its original planned trajectory to the failure controller presented here. The choice of when to do this could substantially impact how effective the failure controller is. Similarly, rigorously determining when the failure controller should give control back to the planner rather than simply always cutting it off after a timeout would likely result in more efficient and robust plans.

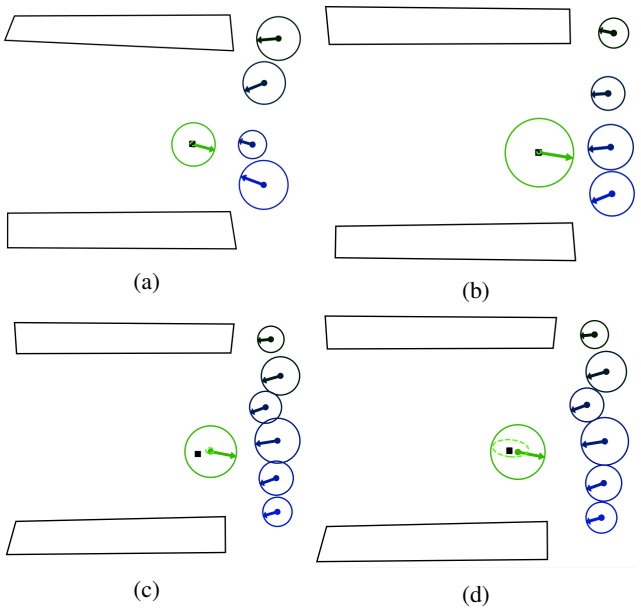


Fig. 5: An example of predictions made by the predictor network for different trajectories. The light green dashed ellipse shows the region where the predictor is 99% confident the robot will be in the next time step. The actual location of the robot in the next time step is denoted by the black square.

VI. Acknowledgements

We would like to thank Rachel Burcin and John Dolan for all of the effort they put into the RISS program to make this experience possible. I would also like to thank Dr. Maxim Likhachev, Ishani Chatterjee, and Tushar Kusnur for their help and support on this project.

References

[1] M. Likhachev and A. Stentz, "Probabilistic planning with clear preferences on missing information," *Lab Papers (GRASP)*, p. 25, 2008.

[2] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *2013 European Conference on Mobile Robots*. IEEE, 2013, pp. 331–336.

[3] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *arXiv preprint arXiv:1809.08835*, 2018.

[4] P. Trautman, J. Ma, R. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation in: Robotics and automation (icra)," in *2013 IEEE International Conference On*, 2013, pp. 2153–2160.

[5] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[8] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[9] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha, "Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation," 2011.

[10] S. Stvel, "Python rvo2," https://github.com/sybrenstuvel/Python-RVO2/, 2017.

[11] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1688–1694.

[14] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1341–1346.

Multitask Learning combining Detection, Segmentation, Tracking and Forecasting

Vivek Roy[1], Luis E. Navarro-Serment[2], Martial Hebert[3]

***Abstract*— Object detection, segmentation, tracking and motion forecasting are fundamental tasks for robotic applications in dynamic scenarios, such as self driving cars planning their motion, or industrial robot arms. But in the perception community they have mostly been seen as independent tasks. Recent works have tried to combine detection and segmentation with tracking, claiming that the three tasks are fundamentally similar and can increase their accuracy if done using one single end-to-end trained network. Motion forecasting and tracking are two related problems demanding temporal consistency; thus approaching them with a single architecture is mutually beneficial. This paper describes a proposed method that combines the tasks of detection, segmentation, tracking and forecasting, approaching them with one single temporally consistent network. Experimental results on the KITTI MOTS dataset show a decrease in false negatives and an overall improvement in tracking accuracy.**

I. Introduction

In the context of static images, object detection and semantic segmentation are high-level tasks that pave the path towards complete scene understanding. In the case of videos, the tasks of tracking and forecasting become fundamentally important for image sequence understanding. The four tasks together can contribute greatly to the vision and perception community by helping us to better understand the world around.

Detection and segmentation has for long been done together on static images. Sharing weights between the detection and segmentation modules allows them to capture knowledge not possible individually.

In the case of object detection and tracking in videos, approaches have mostly used detection as a first step, followed by post-processing methods such as applying a tracker to propagate detection scores over time, known as tracking by detection. Drifting is an inherent problem with visual trackers. Tracking by detection mitigates this problem resetting the tracker to the detection. On the other hand the tracker can help the detector be more confident in localization.

Motion forecasting is tracking in the future. It is typically done by using a network trained on the location of the object in multiple past frames to give the location in future frames. The expectation is that the network will be able to capture the motion model of the concerned object and predict its motion.

1 Vivek Roy is a student of the Department of Computer Science and Engineering at Jadavpur University, Kolkata, India. vivek@vivekroy.com

[2,3]Luis E. Navarro-Serment and Martial Hebert are with the Robotics Institute at Carnegie Mellon University. luisn@andrew.cmu.edu; mhebert@andrew.cmu.edu

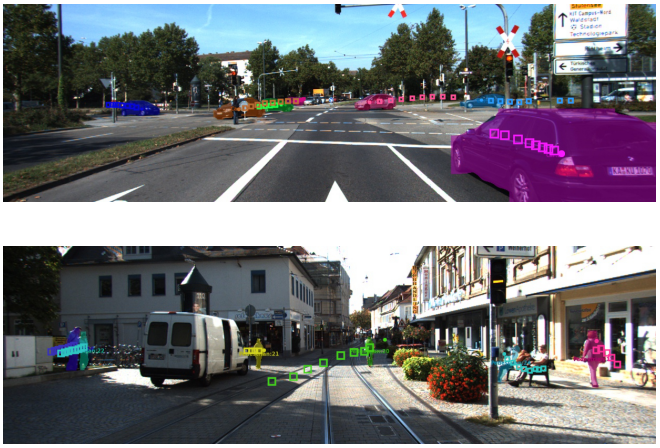


Fig. 1: Sample visualizations showing segmentation masks as well as forecasts. Colored squares represent the location of the centroid in the past frames. Solid circles represent the predicted centroid positions.

Multi-task learning approaches have numerous times proven that sharing weights between related tasks can help make individual tasks more robust. We use the idea of multi-task learning and argue that the tracking and forecasting modules convey the temporal context that the detection and segmentation modules can further utilise to refine their outputs. We use 3D convolutions to capture spatio-temporal information which is then used to perform all four tasks. Leveraging tracking and prediction information can reduce detection false negatives when dealing with occluded or far away objects. False positives can also be reduced by accumulating evidence over time.

In this paper we extend multi-object tracking and segmentation (MOTS) [1] by adding a forecasting module to their architecture. We further use the output of the forecasting module to boost the region proposals around the forecasts, preventing the non-maximum suppression (NMS) step from removing those proposals. The image features for the proposals and their positions in the coordinate system of the image are then passed to the various heads. Similar to Mask RCNN [2] the localization, classification and segmentation modules work on the image features, but on the full stack of features instead of just one feature frame at a time. The association and forecast heads work towards tracking and forecasting respectively.

We use the weights published by [1]. The forecasting

module is further trained on the KITTI and MOT Challenge datasets. We demonstrate the effectiveness of our approach comparing it to [1].

II. Related Work

Detection: Over the last few years, many methods have been deployed that use convolutional neural networks for object detection and localization. These typically work with RGB images in a two-stage fashion [2], [3], [4] – the first stage is dedicated to a region proposal network (RPN) to generate regions of interest (RoIs) where potential objects are located. In the second stage final bounding box locations are predicted from average-pooled features over the proposed RoIs. This two-stage approach eliminates the need for multiple runs of a sliding window object detector with different sizes and aspect ratios, which are computationally heavy.

Detection on videos is accomplished by sampling a video into a sequence of image frames and running a per-image object detection network on a per-frame basis. The detector has no temporal context to consider when detecting the objects – every frame is considered independent of other frames.

Segmentation: In recent years, detection and localization have often been done in tandem with semantic segmentation – the idea of multitask learning claims that similar tasks when done together helps capture more meaningful features than individually possible. Sharing weights between the tasks of detection and segmentation was popularised by Fast-RCNN [5] and has since been adopted by numerous architectures evolving over the years. Instance level semantic segmentation with the introduction of Mask-RCNN [2] has shown to be a simple yet effective way of guiding the network in learning meaningful features.

Tracking: The task of visual object tracking requires following an object which has been marked in the first frame of an image sequence. Works like [6] correlate visual cues between image frames to perform class-less object tracking. Recent works like [7], [8] solve the problem with the help of object detectors, known as *tracking by detection*. The task of tracking is boiled down to associating detections across time – an object detector is used to get per frame detections which are then matched using class information or, in case of multi-instance tracking, heuristics such as distance and intersection-over-union (IoU). MOTS [1] extends on the Mask-RCNN architecture by introducing an association head which is trained to learn instance identification vectors in high dimensional space to be able to associate detections across time.

Forecasting: Related to tracking, motion forecasting is an important temporal task that uses the past location of the object to anticipate its future path. [9] have tried to learn the motion model of the objects from the past motion and forecasts the motion. [10] [11] used LSTMs of some form to model interactions, while [12] used game theoretical concepts to do so. SocialGAN [**?**] uses generative adversarial networks to generate all possible "socially acceptable" trajectories.

	Detection	Segmentation	Tracking	Forecasting
Mask RCNN [2]	✓	✓	-	-
People Tracking by Detection [7]	✓	-	✓	-
Track-RNN [8]	✓	-	✓	-
Motion prediction for people-tracking [7]	-	-	✓	✓
MOTS [1]	✓	✓	✓	-
Ours	✓	✓	✓	✓

TABLE I: Different papers have combined tasks in different ways. Our approach tries to combine all of the tasks into one network.

Multi-task learning: Detection and segmentation has been combined by [2], [4], [5] . Detection and tracking has been combined by [7], [8]. [9] combines tracking and motion forecasting. [1] combines detection and segmentation with tracking.

Different from all the above works, the proposed approach combines all four tasks into a single weight-sharing network as shown in table I. Being able to take advantage of temporal information should improve our ability in non-temporal tasks as well.

III. Approach

In this work, we combine the tasks of detection, segmentation and mask generation with the temporally consistent tasks of tracking and motion forecasting. Towards this goal, we build on top of the architecture proposed by [1] which in turn builds on top of the popular Mask-RCNN [2] architecture by adding an association head to Mask-RCNN. The association head is used to learn an association vector which clusters instances of the same object closer together and different instances of the same or different classes into farther clusters in high dimensional space. Taking a similar approach of having 3D Convolutions to learn temporal context into the image features and adding heads to the Mask-RCNN architecture, we add a forecasting head to learn the motion model across time and predict the location in the frame following the input batch. These predictions are employed to boost the scores of the region proposals in the next frame. Boosting the RPN's scores biases the detector making it give more attention to the

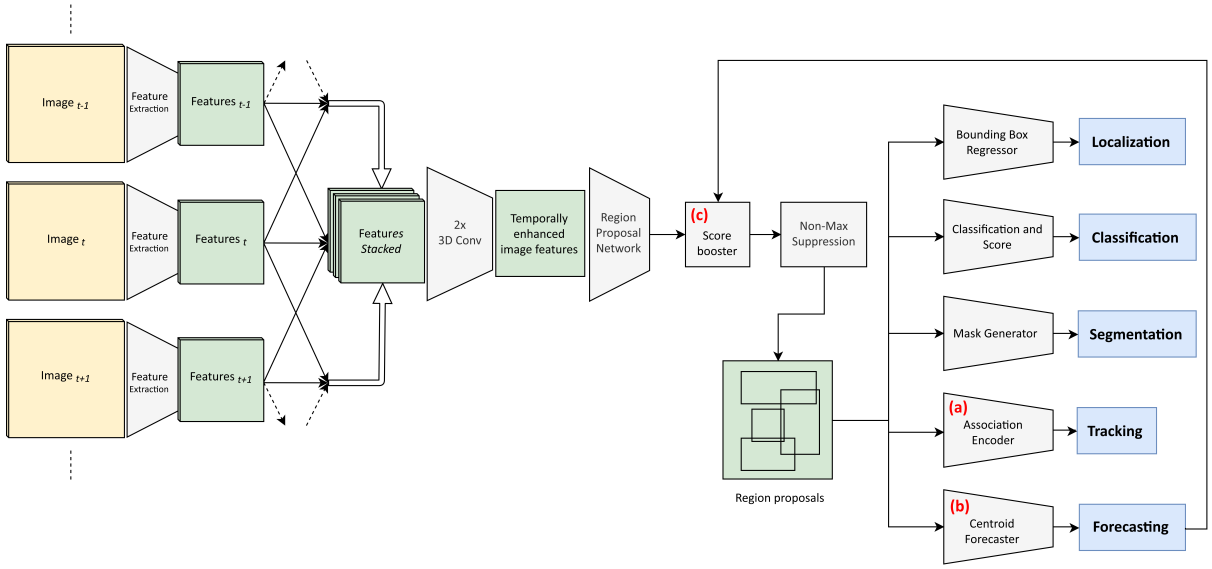


Fig. 2: A high-level diagram of our system architecture.

neighbourhood of the location of the predicted center under the assumption that a good prediction module will greatly help restrict the region of possible appearance of the object under consideration in the next frame. This mechanism helps boost detection and segmentation accuracy by providing temporal context augmented to the otherwise independent video frames.

Our input representation is a 3D tensor of video frames across time. 3D convolutions are employed to learn image features. We call these temporally enhanced as they augment temporal context from the video frames into the learnt image features. These enhanced features are passed to a RPN to generate per-frame region proposals while also taking into account the temporal information. These region proposals then pass through a boosting network which essentially is a fully-connected layer with input being the predictions for the current set of frames, the region proposals and their corresponding scores and output being the captured motion model boosted region proposals. The image features are then cropped on the proposals obtained after running non-max suppression on the boosted region proposals. Cropped region proposal are sent to the different heads for detection, segmentation and mask-generation as well as to the association and prediction heads for associating the regions across time and generate the centroid prediction for the frame following the input batch frames respectively.

The motion forecasting head is modelled as a fully-connected layer with the input being the centroids of the region proposals across the input batch and output being the centroid of each of them in the following frame. We use a weighted smooth $L1$ loss for training the module given by:

$$smooth_{L1}(\hat{x}, x) = \begin{cases} \frac{1}{2}(\hat{x} - x)^2 & \text{if } |\hat{x} - x| < 1 \\ |\hat{x} - x| - \frac{1}{2} & \text{otherwise} \end{cases} \quad (1)$$

where x is the output of the model and $\hat{x}$ is the true label corresponding to the output.

IV. Results

[1] proposes three new metrics for simultaneous detection, tracking and segmentation – MOTSA, MOTSP and sMOTSA. We use these metrics to evaluate our work. They are defined as follows:

Multi-object tracking and segmentation accuracy (MOTSA) is given by

$$\begin{aligned} MOTSA &= 1 - \frac{|FN| + |FP| + |IDS|}{|M|} \\ &= \frac{|TP| - |FP| - |IDS|}{|M|} \end{aligned} \quad (2)$$

where FN is the number of false negatives, FP is the number of false positives, IDS is the number of ID switches, TP is the number of true positives and M is the total set of all masks. Each object is assigned an ID. If an object is not detected for a significant amount of frames, a new ID is assigned to the same object. This is what we refer to as ID switching.

Multi-object tracking and segmentation precision (MOTSP) is given by

$$MOTSP = \frac{\widetilde{TP}}{|TP|} \quad (3)$$

where $\widetilde{TP}$ is the soft version of number of true positives given by

$$\widetilde{TP} = \sum_{h \in TP} IoU(h, c(h)) \quad (4)$$

Finally, soft multi-object tracking and segmentation accuracy is given by

$$sMOTSA = \frac{\widetilde{TP} - |FP| - |IDS|}{|M|} \quad (5)$$

Table II compares the performance of our model in comparison to MOTS in the three metrics proposed in that paper. Table III shows our performance in the number of False Positives (FP), False Negatives (FN) and IDS (ID-switches). Table IV shows the number of Mostly Tracked (MT), Partially Tracked (PT) and Mostly Lost (ML) of our model.

	MOTSA		MOTSP		sMOTSA	
	Car	Ped	Car	Ped	Car	Ped
TrackR-CNN	87.8	65.1	87.2	75.7	76.2	46.8
Ours (without boosting)	88.2	64.8	87.5	75.1	77.5	46.1
Ours (with boosting)	**91.2**	**66.3**	**88.5**	**76.3**	**78.6**	**47.0**
GT Boxes	95.3	71.1	86.9	77.4	82.5	50.0

TABLE II: Quantitative results in the three metrics on the KITTI MOTS dataset.

	FP		FN		IDS	
	Car	Ped	Car	Ped	Car	Ped
TrackR-CNN	**134**	**267**	753	822	93	78
Ours (without boosting)	176	317	**370**	650	53	91
Ours (with boosting)	151	282	377	**647**	**46**	**65**

TABLE III: Number of false positives (FP), false negatives (FN) and identity switches (IDS) in the KITTI MOTS dataset.

	MT		PT		ML	
	Car	Ped	Car	Ped	Car	Ped
TrackR-CNN	87.4	61.8	11.3	32.4	1.3	5.9
Ours (without boosting)	88.1	62.1	10.6	32.0	1.3	5.9
Ours (with boosting)	90.2	63.1	8.6	31.1	1.2	5.8

TABLE IV: Percentage of mostly tracked (MT), partially tracked (PT) and mostly lost (ML) in KITTI MOTS dataset.

The number of false negatives have greatly been reduced while also keeping the number of false positives in control. We see a significant increase in the case of cars, but not as much improvement in the case of pedestrians. This is because large objects like cars do not maneuver nearly as much or as unpredictably as pedestrians do; consequently, predicting their motion is less challenging than that of humans.

Object identities are better maintained. This can be attributed to the fact that the forecasting module in combination with the NMS score booster helps to detect the objects which were skipped by previous models. A few undetected frames between two detections can lead to an object switch. The forecaster reduces the number of undetected frames between two detected frames and thus improves the ID switching performance.

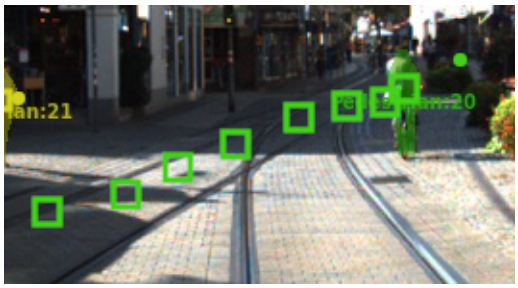


Fig. 3: Here we see a bicyclist cutting across the street and then moving forward. When the person just starts moving forward, the history comprises of centroids of the person cutting across the street and thus the model predicts in line with that. The model gives equal importance to the whole history.

Fig. 4: We can see two people sitting next to each other, but only one of them is detected. The score boosting module boosts the region proposals of the one that was detected and as a side-effect suppressing the region proposals corresponding to the other person. Thus, if an object is not detected in the first few frames, it is not detected at all.

Our model fails to detect more than one object when they are too close. We can attribute this to the NMS score boosting. Hence, there is not much improvement in the *mostly lost* percentage. Figure 4 illustrates the same. Objects undetected in the first frame they appear are not detected ever.

Finally, the model tends to give equal importance to all past frames. Giving more importance to recent history can arguably be better for pedestrians for example in the case of sharp turns and sudden motion. This can be seen in figure 3. But these cases do not arise nearly as much in the case of vehicles. The dataset is biased, containing approximately 2.5 times more vehicles than pedestrians. There are approximately 8 thousand vehicles and 3 thousand 3 hundred pedestrians. As a result, the network learned to give equal importance.

V. Conclusion

Until now people have combined one or two tasks into multi-task learning problems. This work further proves that multi-task learning can help capture information that would otherwise be really difficult for individual modules to figure out. Tracking and forecasting being related tasks can really benefit from each other. The forecasting module can help the

detector localize objects and thus also improving segmentation performance. Thus combining more related tasks keeps improving performance. The more the merrier.

ACKNOWLEDGMENTS

This work was done under the financial support of the Federation of Indian Chambers of Commerce & Industry (FICCI). We would also like to acknowledge Rachel Burcin and Dr. John M. Dolan for their tireless commitment to the Robotics Institute Summer Scholars program.

REFERENCES

[1] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "MOTS: multi-object tracking and segmentation," *CoRR*, vol. abs/1902.03604, 2019. [Online]. Available: http://arxiv.org/abs/1902.03604

[2] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[3] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," *CoRR*, vol. abs/1605.06409, 2016. [Online]. Available: http://arxiv.org/abs/1605.06409

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[5] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[6] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," *CoRR*, vol. abs/1604.01802, 2016. [Online]. Available: http://arxiv.org/abs/1604.01802

[7] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[8] K. Fang, "Track-rnn : Joint detection and tracking using recurrent neural networks," 2016.

[9] F. Broz and G. Gordon, "Better motion prediction for people-tracking," 03 2004.

[10] Y. Xu, Z. Piao, and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[11] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "SR-LSTM: state refinement for LSTM towards pedestrian trajectory prediction," *CoRR*, vol. abs/1903.02793, 2019. [Online]. Available: http://arxiv.org/abs/1903.02793

[12] W. Ma, D. Huang, N. Lee, and K. M. Kitani, "A game-theoretic approach to multi-pedestrian activity forecasting," *CoRR*, vol. abs/1604.01431, 2016. [Online]. Available: http://arxiv.org/abs/1604.01431

Systems Development for Aerial Manipulation Tasks

Andrew Saba[1], Oliver Kroemer[2] and Sebastian Scherer[3]

***Abstract*— Unmanned aerial vehicles (UAVs) have become increasingly capable over the past decade, rising from hobbyist projects to commercial products and industrial tools. As aerial robotics in general develops, UAV systems are increasingly found helping human operators in dangerous situations, such as inspecting infrastructure that is too high or dangerous to access, and entering or flying over burning buildings. However, in a lot of cases, these systems take on the role of mobile eyes for human operators and are not able to manipulate their environment nor navigate without a human aid in these unstructured, dangerous environments. Recent research looks to bridge this gap by developing aerial platforms capable of taking on more aerial manipulation tasks fully- or semi-autonomously. This work covers one such developed system that is robust enough to operate in these environments while also being relatively low-cost and utilizing a number of off-the-shelf sensors.**

I. Introduction

With modern advances in robotics, robots are becoming increasingly capable of performing tasks that were once delegated to humans. Aerial Manipulation tasks, such as infrastructure contact inspection, construction, and search and rescue tasks, are often dangerous for humans. For example, in [1], a 2 degree of freedom manipulator is mounted onto a quad copter for the purpose of performing dynamic contact interactions, such as in contact inspections or object manipulation. The goal is that unmanned aerial vehicles (UAVs) can replace or augment humans in these dangerous situations to reduce injury or even avoid death. However, due to the unstructured and dangerous nature of these environments, several problems arise that hinder the capability of autonomous UAVs.

Aerial manipulation tasks for aerial vehicles are difficult for several reasons. Firstly, any object or surface being manipulated or made contact with can obstruct any onboard sensors. Additionally, the environment might be obstructing the vehicle's sensors through magnetic or other interference, such as poor GPS signal below tall structures. This is especially important if the systems are relying on said sensors for state estimation, for example, for the autonomy is unable to know where it is in the world and may act inappropriately (i.e. cash, fly erratically). Secondly, the act of making contact impacts the dynamics of the vehicle and imparts a force into the system that must be counteracted. Spikes associated with contact can impact controllers and again cause the vehicle to fly erratically as it corrects. Additionally, if the motors and propellers are not capable of providing the thrust response needed to overcome the contacts, then the overall system can become unstable, since now the impulse can never be countered completely. Finally, there are unknown constraints when the system is interacting in unstructured environments. Objects could or could not be compliant, for example.

(a)

(b)

Fig. 1: (a) Left shows the current camera feed, where a user selects a bounding box of interest. Right shows the selected bounding box. Bottom row lets user switch autonomy modes. (b) Output of a CSRT feature tracker, tracking the user-selected bounding box

This works addresses some of these problems, namely sensor obstruction, especially in the case of GPS obstruction, and force impulses impacting the system dynamics. Additionally, this work also demonstrates a process for optimizing an aerial platform for agility and flight time. Agility is important for reacting to unknown constraints or contact impulse forces and a long enough flight time is needed for

[1]Andrew Saba is a senior computer engineering student at the University of Pittsburgh, Swanson School of Engineering in Pittsburgh, Pennsylvania asaba96@pitt.edu

[2]Dr. Oliver Kroemer is an Assistant Professor in the Robotics Institute at Carnegie Mellon University in Pittsburgh, Pennsylvania okroemer@andrew.cmu.edu

[3]Dr. Sebastian Scherer is an Associate Research Professor in the Robotics Institute at Carnegie Mellon University in Pittsburgh, Pennsylvania basti@andrew.cmu.edu;

the platform to be useful to the operator. Finally, the solutions developed in this work utilizes off the shelf components and software packages, where possible, to minimize cost and reduce risk in the event of crashes, since replacement parts are readily available.

II. Hardware Platform

A. System Requirements

Aerial manipulation tasks are demanding and require a rugged, robust, customizable, and powerful platform. The platform must be capable of operating in an outdoor environment under and over various forms of infrastructure (bridges, roadways, towers, etc.). It must be robust enough to make up to thousands of repeated contacts with infrastructure or other objects in its lifetime. The platform must also be able to support various sensors, some of which may be relatively heavy (2 to 5 kilograms). The platform must also be able to accelerate rapidly in the event of safety maneuvers, such as accelerating away from an obstacle that was recently discovered, which can happen if something is dislodged during the contact. Additionally, in order to be useful for an operator, the vehicle must be capable of flying for a relatively long time (at least 15 minutes).

Fig. 2: Top View of the developed UAV system

B. Frame, Motors, and Compute

To solve this problems, a hexarotor platform was developed that uses a Tarot X6 frame, with a 0.96m motor-to-motor diameter and a maximum payload of 7.5kg (see Figure 2). The frame can be size-adjusted for specific tasks. Propulsion is achieved with six KDE Direct KDE4215XF-465 brushless motors and KDEXF-UAS55HVC electronic speed controllers (ESC). The flight controller is a Pixhawk 2 Cube running a modified version of the PX4 flight stack [2]. Compute and sensor payload can be seen in Figure 5. Onboard computation is achieved using an Intel NUC computer. An Intel T265 RealSense Camera is used to provide visual odometry information and a D435 RealSense Camera is used for RGB-D imaging, and an SF30 Laser Rangefinder is used to provide accurate height estimation.

The frame is relatively large (96cm motor to motor), allowing for a large payload (over 5kg) and numerous customization options (i.e. ability to mount different arms, end effectors, or various sensors. Additionally, the motors and propellers chosen allow for about 20 minutes of flight, while also providing almost 29kg of peak thrust. Additionally, by utilizing the sensors listed above, reliable state estimation in a GPS-denied environment was achieved at a relatively low-cost (less than 500 dollars) and with little addition weight or power requirements.

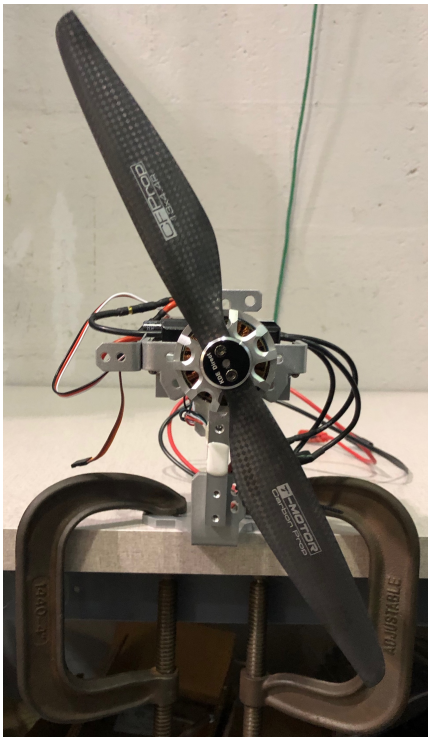


Fig. 3: Thrust stand setup

C. Thrust Testing

To ensure that the motors and propellers selected could provide the necessary flight time, static and dynamic thrust tests were conducted with a thrust stand (see Figure 3). This builds on previous works, such as in [3], which used a thrust stand to characterize various motor and propeller configurations and determine the most optimal configurations. By thrust testing, experimental data is collected to guide engineering decisions and provide guidance on the best propulsion designs.

The motor and several propeller comparisons were made. Figure 4 has a comparison between two propeller sizes that were considered, with data collected with the same motor (KDE 4215XF) with a static thrust sweep test, which involves ramping thrust from 0% to 100%, holding, and ramping back down to 0. Propellers were both T-Motor polished carbon fiber, one being 13” long and the other 12”.

On the left of Figure 4 is a graph of the current measured, in amperes, versus the thrust generated, in Newtons. On the right is a graph of the electrical power measured (voltage times current draw) versus thrust generated, again in Newtons. There are several important points to note here when picking the best propeller for a motors.

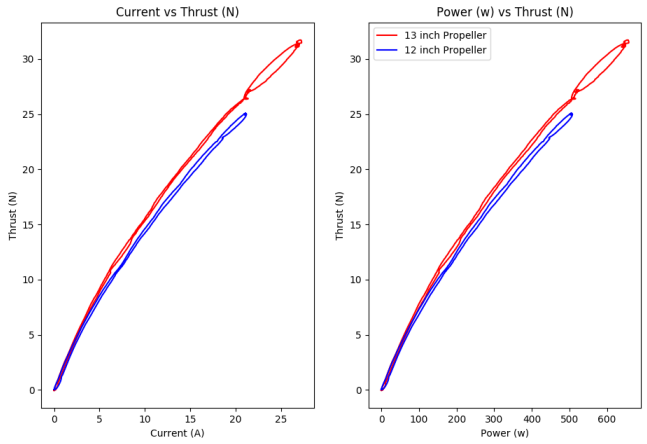


Fig. 4: Thrust data collected for two propeller sizes

Firstly, the propeller cannot cause the motor to draw more current than what it is rated for. In this case, neither propeller does so. Secondly, the best propeller generates more thrust with less current and power drawn. In this particular comparison, the 13 inch propeller generated more thrust for the same amount of current and power. Finally, the best propeller generates a peak thrust that is at least one and a half to two times the mass of the vehicle and payload being carried. For example, if the vehicle and payload weight 6kgs, then the total thrust generated by all the motors and propellers should be at least 9 to 12kgs. In this particular case, the 12 inch propeller generated about 7-8 N fewer of thrust when compared to the 13 inch. In this particular comparison, the winner is clear; however, there are cases where peak thrust might need to be traded for efficiency, depending on the propeller and motor choices.

The result of this work is a vehicle capable of flying for roughly 20 minutes while carrying a one kilogram payload. Additionally, as mentioned earlier, the platform can generate over 29kgs of thrust, resulting in very agile movement for such a large vehicle. Finally, the components used are relatively cheap and easy to replace, as mostly off the shelf components were utilized.

III. State Estimation

In order to robustly control the vehicle when manipulating the environment, localization in the world needs to be robust and locally accurate. In the long-term, some positional drift is acceptable. Visual servoing techniques correct for long-term drift by estimating the target's position relative to the body and transforming the target point into the map (global) frame. However, an accurate state estimate is needed to precisely control the vehicle to reach the target position.

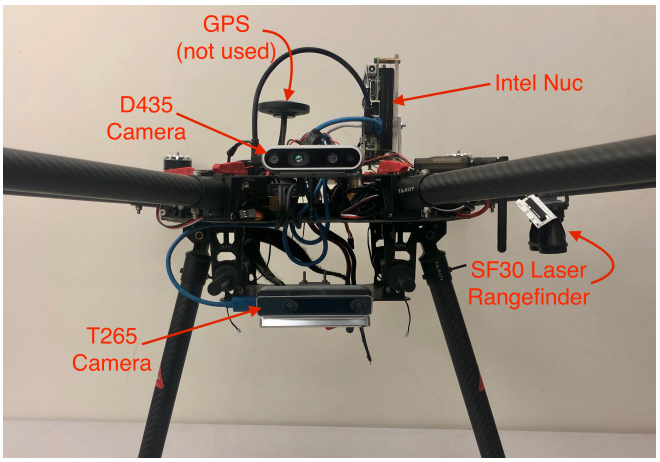


Fig. 5: Sensor and compute payload

Bridges, buildings, and other large structures often block GPS signals and interfere with other sensors, rendering most off-the-shelf localization solutions inoperable for aerial manipulation platforms. To solve this problem, an Intel T265 RealSense tracking camera is utilized, along with an SF30 Laser Rangefinder and the onboard IMU (see Figure 5). The T265 RealSense tracking camera utilizes a proprietary SLAM algorithm to generate a 6 degree of freedom pose.

Readings from the onboard sensors are fused in a 15 degree of freedom Extended Kalman Filter (EKF) node provided by Robot Localization [4]. Utilizing the flexibility of the filter, the tracking camera was utilized for position and velocity in the horizontal plane and for absolute orientation. The SF30 provided an accurate measure (within +-10cm) of the vehicle's altitude. Finally, the IMU provided linear and angular acceleration and orientation measurements. This particular configuration provided the best performance in test flights.

A few alternative approaches to the problem were attempted, all of which using the T265 tracking camera. Originally, the tracking camera's velocity estimate was fused with the rangefinder and IMU. However, through field testing it was found that fusing velocities alone caused significant drift (greater than roughly 20cm positional drift within less than 5 minutes of flight).

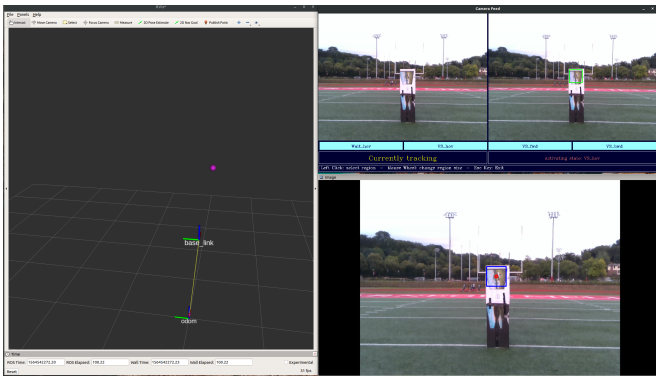


Fig. 6: Thrust data collected for two propeller sizes

Additionally, it was discovered that no absolute orientation could be measured on-site, due to high magnetic interference with the onboard magnetometer from large nearby metal structures. As a result, it was decided to fuse the absolute position and orientation estimates provided by the T265. This approach eliminated most of the drift while maintaining a locally accurate estimate.

IV. Field Testing

Overall, reliable localization and state estimation was achieved utilizing the onboard sensor suite, despite the lack of conventional means of localization, such as GPS. This was demonstrated and tested on-site at the Charles Anderson Bridge in Schenley Park, Pittsburgh, PA, with manual flights and on the Carnegie Mellon University football field with autonomous flights. In Figure 6, on the left is a visualization of where the drone thinks it is in the world and the target point where the vehicle is to make contact (purple sphere). On the right is an image from an onboard region of interest tracker that is tracking the region the vehicle is to make contact with. The localization is consistent with where the drone actually was in real life. Additionally, the state estimate worked reliably enough for the vehicle to maintain its position while waiting for a region of interest to servo to and for using to estimate where the target is in the world and moving towards that point.

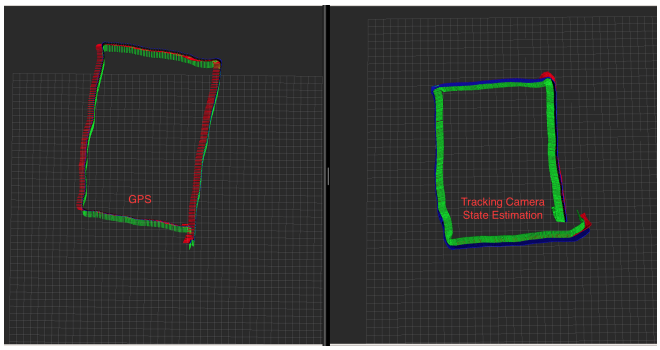


Fig. 7: Side by side comparison of raw GPS readings and state estimation

Data was also collected during additional field testing on the Carnegie Mellon University football field, where the vehicle was autonomously flying in a 21m by 15m rectangle while utilizing GPS. Sensor data was later run through the developed state estimation systems and compared against GPS in 7. The largest two takeaways from this comparison are that the state estimate drifted by less than 1 meter after flying for over 80m, meaning that the final estimate in flight was within one meter of the starting point, which is roughly where the vehicle landed. Secondly, the GPS estimates for distance traveled down each side and back is consistent and within a meter or two of what the state estimate is outputting.

This all indicates that the estimate is consistent with GPS, even after such a long and fast flight. Additionally, this level of accuracy and consistency is more than acceptable for aerial manipulation tasks, where often the autonomy system is able to re-orient itself relative to the target despite its state estimate drifting using techniques such as visual servoing . Finally, the developed system, unlike other off the shelf solutions that depend on GPS alone, can work in significantly more environments.

Fig. 8: Fully actuated hexarotor with tilted propellers

V. Conclusion and Future Work

Thrust testing has been used to develop smaller platforms than the one in this work as well as a fully actuated hexarotor platform (Figure 8) that is capable of moving in 6 degrees of freedom without rolling or pitching. The state estimation and use of the T265 camera are also being used on these smaller vehicles, demonstrating the transfer-ability of the developed work. The hope is to expand the use of these systems, build on top of them, and provide a set of base platforms for more projects to come.

The systems developed in this work are robust, powerful, able to work in unstructured, GPS-denied environment, and utilize off-the-shelf solutions. These systems will be utilized in numerous projects, including for infrastructure inspection, construction, and fire fighting. This work sought to develop a robust platform that multiple projects will be able to utilize and build on top of.

Acknowledgment

The author would like to thank Professors Sebastian Scherer and Oliver Kroemer for the opportunity to work on this project. Additionally, the author would like to thank Azarakhsh Keipour for all of his guidance and help on the project. Additionally, the author is thankful for Anish Bhattacharya and Lorenz Stangier for their help and contributions to this project and Greg Armstrong and Nora Kazour for all of their assistance and help over the course of the summer. The author is especially thankful to Rachel Burcin and Dr. John Dolan for a wonderful summer experience in the RISS Program. Finally, thanks to the National Science Foundation for financially supporting this work through a Research Experience for Undergraduates grant.

REFERENCES

[1] T. Bartelds, A. Capra, S. Hamaza, S. Stramigioli, and M. Fumagalli, "Compliant aerial manipulators: Toward a new generation of aerial robotic workers," *IEEE Robotics and Automation Letters*, vol. 1, pp. 1–1, 01 2016.

[2] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6235–6240.

[3] J. Bell, M. Brazinskas, and S. Prior, "Optimizing performance variables for small unmanned aerial vehicle co-axial rotor systems," in *Engineering Psychology and Cognitive Ergonomics*, D. Harris, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 494–503.

[4] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

Motion Planning for Urban Driving Including Evasive Maneuvers Using Iterative LQR

Het Shah[1], Yanjun Pan[2] and John Dolan[3]

***Abstract*— There exist a lot of challenges in motion planning for autonomous driving, especially in the case of sudden motion of dynamic obstacles. A real-time implementation of spatial and temporal planning that can handle nonlinear vehicle model and dynamic obstacle avoidance is needed. Iterative Linear Quadratic Regulator (iLQR) solves predictive optimal control with non-linear systems. In this paper, we use Constrained Iterative Linear Quadratic Regulator for motion planning for urban areas in highly dynamic environments and deal with scenarios such as sudden movement of a pedestrian/animal/other vehicles on the road which usually leads to an evasive maneuver. State dependent auto tuning of cost weights has been proposed to deal with multiple scenarios.**

I. INTRODUCTION

It is estimated that self-driving cars will have a significant impact on the road transportation in the near future, and there is currently a lot of related research going on in academia as well as industry. However, they need to deal with certain emergency situations caused by other vehicles, pedestrians and wildlife. Motion Planning and Control is one of the most important modules in autonomous vehicles which takes information of the outside environment, plans a trajectory to execute and give commands to the actuators of the vehicle. Figueiredo et al. [1] shows challenges for motion planning due to factors such as steep roads, curves, surface uncertainties, etc.

Path planning approaches can be divided into three categories: 1) Search-based planning 2) Sampling-based planning 3) Optimization-based planning [2]. Search based planners like A*, D*, etc. give globally optimal solutions but they are inefficient when dealing with dynamic obstacles. Also, finding a good heuristic function is a challenge. Sampling-based planners result in sub-optimal solutions and are inefficient in collision avoidance. Also, as these both are performed in discrete space, the trajectories are not smooth and computational efficiency is affected. Optimization-based methods formulate the motion planning as an optimization problem and result in continuous and spatiotemporal trajectories.

A. Related Works

The problem of trajectory planning and tracking through control of vehicle steering and acceleration has been widely studied over years. Trajectory optimization can be formulated as a quadratically constrained quadratic program [3]. Ziegler et al. [4] used Sequential Quadratic Programming (SQP) to solve the nonlinear and non-convex problem, but the computation time is around 0.5s, which is not suitable for real-time implementation. Xu et al. [5] proposed an efficient real-time motion planner with trajectory optimization that discretizes the plan space and selects the best trajectory based on a cost function.

Control methods such as LQR and Model Predictive Control (MPC) have also been used for this purpose. Two layer MPC which reduces the computation cost as compared to one layer MPC has been proposed for motion planning with obstacle avoidance [6]. Arab et al. [7] presented a Sparse-RRT* and nonlinear MPC based motion planner for aggressive maneuvers. Ji et al. [8] presents a path planning and tracking framework using 3D potential fields and multi-constrained MPC. In [9] an nonlinear MPC has been implemented to stabilize the vehicle along a desired path, but there has been a trade off between vehicle speed and the prediction horizon. As a result, heavy computation still stands as a barrier for nonlinear MPC. Iterative linear quadratic regulator is an optimization-based method for nonlinear systems with lower computation time. However, very few works have considered constraints in iLQR. The control-limited differential dynamic programming (DDP) was proposed in [10], deals with constraints of upper and lower bound on control inputs. Extended LQR [11] - [12] works with collision avoidance for circular obstacles, penalizing the distance from the center of the obstacle.

B. Our Contribution

This paper presents a motion planning approach for autonomous vehicles which can generate the trajectory and plan the control inputs which can track the trajectory. The majority of road accidents are caused by vehicle collision. The aim of the collision avoidance system is to avoid imminent accidents using longitudinal (emergency braking) or lateral control (active steering). However, emergency braking is not possible for every situation as there might not be enough distance or time to stop before an obstacle. Our approach focuses on integration of emergency obstacle avoidance scenarios.

Iterative Linear Quadratic Regulator (iLQR) [13], an algorithm based on Linear Quadratic Regulator is used to solve unconstrained optimization problems with nonlinear system dynamics. However, it has not seen much appearance in the field of motion planning for autonomous vehicles as it does

*This work was funded by Federation of Indian Chambers of Commerce and Industry (FICCI)

[1]Het Shah is a 2019 Robotics Institute Summer Scholar and a student at Indian Institute of Technology, Kharagpur hetshah369@iitkgp.ac.in

[2]Yanjun Pan is a master student at Carnegie Mellon University yanjunp@andrew.cmu.edu

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

not deal with the constraints and constraints are inevitable for autonomous driving. But, the authors in [14] have shown that constraints can be embedded in the cost function and can be used for the motion planning problem but does not deal with obstacles moving in lateral direction. It deals with dynamic obstacles just by re-planning and using a closed loop control. Also, tuning a set of universal cost weights is difficult which works efficiently with multiple different scenarios. In this paper, we have used constrained iLQR to specifically deal with scenarios of dynamic sudden obstacles like sudden animal crossing the vehicle and other vehicles cutting in the ego vehicle's lane. A constant acceleration model of dynamic obstacles is considered while adding them in the cost function which allows the ego vehicle to efficiently make evasive maneuvers. We also propose a state dependent tuning of cost weights which can handle multiple scenarios.

II. System Description

A. System Dynamics

The kinematic bicycle model as shown in Fig. 1 is used in this paper. The kinematic model considers a single pair of wheels at the center of front and rear axles instead of left and right pairs. The zero lateral slip assumption is taken. The state consists of $x = [p_x, p_y, v, \theta]^T$ and the control input is $u = [\dot{v}, \dot{\theta}]$, where p_x, p_y represent the position coordinates, v is the vehicle velocity, θ is the orientation of the vehicle, $\dot{v}$ and $\dot{\theta}$ are the acceleration and yaw rate of the vehicle respectively. The equation 1 represents the vehicle dynamics.

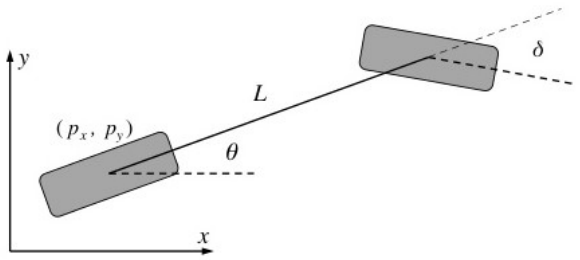


Fig. 1: Vehicle Kinematic Model

$$\dot{x} = \begin{bmatrix} vcos\theta \\ vsin\theta \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\theta} \end{bmatrix} = g(x, u) \tag{1}$$

Since, we would be dealing with the discrete time domain, this model is transformed to discrete time form as described in equation 2, where T_s is the sampling time.

$$x_{k+1} = \begin{bmatrix} p_{x,k} + cos\theta_k(v_k T_s + \frac{1}{2}\dot{v_k} T_s^2) \\ p_{y,k} + sin\theta_k(v_k T_s + \frac{1}{2}\dot{v_k} T_s^2) \\ v_k + \dot{v_k} T_s \\ \theta_k + \dot{\theta_k} T_s \end{bmatrix} = f(x_k, u_k) \tag{2}$$

B. Constraints

There are some essential constraints that are to be obeyed by the autonomous vehicle while driving on road. Here are the 3 constraints that are to be considered.

1) Control Input Constraints: The control inputs i.e. acceleration and yaw rate are bounded by some upper and lower limits. The acceleration is bounded due to the engine power and braking torques for tires. The yaw rate is bounded due to the constraints in steering angle and vehicle longitudinal velocity.

$$a_{min} < a < a_{max} \tag{3}$$

$$\delta_{min} < \delta < \delta_{max} \tag{4}$$

2) Position Constraints: The vehicle needs to be bounded between the specific lanes and avoid the curbs at the side of the road. Also, there are a lot of scenarios which have boundary constraints in longitudinal direction [14], i.e. stop sign scenario or a car following scenario.

3) Obstacle Avoidance: The obstacle avoidance deals with static and dynamic obstacles. Two type of dynamic obstacles are considered: 1. Other cars moving along the ego vehicle in the same or adjacent lane 2. Obstacles such as human or wildlife crossing the road in front of ego vehicle. As it is difficult to deal with convex obstacle (e.g. rectangle representing a car), all the obstacles are modeled as ellipse. The authors in [14] has also shown more advantages of using ellipse for representation of obstacles.

The long and short axis are formulated according to the obstacle's shape, current velocity and acceleration. The long axis of the ellipse may be set as $a = (l + v_x t_{safe} + s_{safe})$ and $b = (w + v_y t_{safe} + s_{safe})$, where l and w are the length and width of obstacle, v_x and v_y are the longitudinal and lateral velocity of obstacle, t_{safe} and s_{safe} are the safe headway and safety margin. A constant vehicle acceleration model is assumed for the obstacles, so v_x and v_y of the future time step should be considered for the cost function.

III. Overview of Method

A. iLQR

The iterative LQR algorithm is used to solve for non linear systems. The problem formulated is as follows

$$\min_{\substack{u_0,\dots,u_{N-1} \\ x_0,\dots,x_N}} J = \frac{1}{2} x_N^T Q_N x_N + x_N^T p_N x_N + q_N + \sum_{t=0}^{N-1} (\frac{1}{2} x_t^T Q_t x_t + x_k^T p + \frac{1}{2} u_t^T R_t u_t + u_k^T r + q) + x_H^T P_H x_H \tag{5}$$

$$s.t. \quad x_{k+1} = f(x_k, u_k), \quad k = 0, 1, ..., N-1 \tag{6}$$

$$x_0 = x^{start} \tag{7}$$

where Q_n, p_n, q_n, Q, p, R, r, q are the parameters describing the cost function. The dynamic function in the equation (6) can be non linear.

Following are the steps to solve the above problem:

1) Start with a feasible initial guess $\hat{u_t} = u_0$ and obtain $\hat{x_t} = x_0$ using equation 6.
2) Calculate the derivatives of dynamic and cost function about the previous trajectory.

$$F_t = \nabla_{x_t,u_t} f(\hat{x_t}, \hat{u_t}) \tag{8}$$
$$j_t = \nabla_{x_t,u_t} J(\hat{x_t}, \hat{u_t}) \tag{9}$$
$$J_t = \nabla^2_{x_t,u_t} J(\hat{x_t}, \hat{u_t}) \tag{10}$$

3) Run LQR backward pass on state $\delta x_t = x_t - \hat{x_t}$ and action $\delta u_t = u_t - \hat{u_t}$.
4) Run forward pass with real nonlinear dynamics.
5) Update $\hat{x_t}$ and $\hat{u_t}$ based on states and actions in forward pass.
6) Iterate the whole process until the cost value converges.

B. Cost Function

The cost function for the optimization problem contains the following terms:

$$J = \sum_{t=0}^{N} (c_k^{control} + c_k^{\Delta control} + c_k^{ref} + c_k^{vel} + c_k^{constraints}) \tag{11}$$

The description of the computation of each term is given below:

1) Control input: The control inputs and changes in control inputs are penalized to avoid excess use and sudden change of acceleration and yaw rate giving the passengers a comfortable ride.

$$c_k^{control} = \omega_{acc} u_k^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u_k + \omega_{yaw} u_k^T \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} u_k \tag{12}$$

$$c_k^{\Delta control} = \omega_{\Delta acc} (u_k - u_{k-1})^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} (u_k - u_{k-1}) + \omega_{\Delta yaw} (u_k - u_{k-1})^T \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} (u_k - u_{k-1}) \tag{13}$$

2) Reference Offset: This term penalizes the distance from the reference. The distance from the nearest point on the reference path is considered.
3) Velocity Difference: The velocity difference term is added to achieve the desired velocity along the trajectory.

$$c_k^{vel} = \omega_{vel} (v_{des} - v) \tag{14}$$

4) Constraints: The constraints equations are embedded in the iLQR as a cost function to the optimization problem. It has been shown that constraints can be effectively added as barrier function in the cost function [13]. The constraint function f_k^x can be converted to the barrier function b_k^x using the equation 15. The barrier function can be quadratized by using Taylor series expansion as shown in the equation 16.

$$b_k^x = q_1 exp(q_2 f_k^x) \tag{15}$$

$$b_k^x(x_k + \delta x_k) \approx \delta x_k^T \nabla^2 b_k^x(x_k) \delta x_k + \delta x_k^T \nabla b_k^x(x_k) + b_k^x(x_k) \tag{16}$$

By applying chain rule, we have:

$$\nabla b_k^x(x_k) = q_1 q_2 exp(q_2 c_k^x(x_k)) \tag{17}$$
$$\nabla^2 b_k^x(x_k) = q_1 q_2^2 exp(q_2 c_k^x(x_k))(\nabla c_k^x(x_k))(\nabla c_k^x(x_k))^T \tag{18}$$

C. Cost Weights

Tuning a set of universal cost weight that gives optimal results for all the scenarios is difficult. Keeping the cost weights like c_{vel} and c_{ref} same for various scenarios does not work efficiently. We come up with a state dependent tuning of cost weights which can be generalized for more scenarios.

The higher the value of c_{vel} is, the more encouraged the ego vehicle is to track the reference path with the desired velocity, whereas the higher value of c_{ref} is, the more encouraged the ego vehicle is to follow the path with minimum reference offset. So, with obstacles far away from ego vehicle, the ego vehicle do not need to perform an obstacle avoidance task, and therby track the trajectory. Whereas when the distance between the obstacle and ego vehicle is decreasing it is desired that the vehicle should overtake or perform avoidance maneuver depending upon the current velocity and states. In this case the reference tracking can be relaxed. A linear function for both cost weights is used as shown in the equation 19-20. Both weights are clipped at some minimum and maximum values tuned at extreme cases.

$$c_{vel} = c_{v1} - d_{obs} c_{v2} \tag{19}$$
$$c_{ref} = c_{r1} + d_{obs} c_{r2} \tag{20}$$

IV. Experimental Results

The experiments were carried out and simulated on MATLAB. Results for various test cases for urban driving are shown. Especially the effect of tuning weights have been shown for normal scenarios and scenarios with evasive maneuvers.

A. Static Obstacles

The figure 2 shows the ego vehicle avoiding the static obstacles on the way. The sequence of black boxes shows the ego vehicle at each time step and the blue path shows the trajectory followed by the vehicle

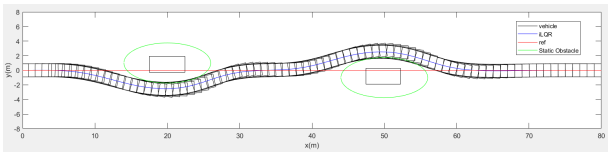
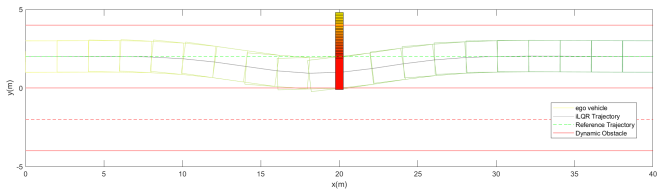


Fig. 2: Static obstacle avoidance

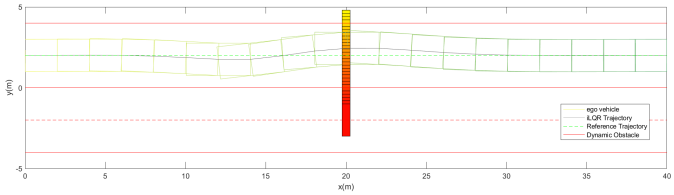
B. Dynamic Obstacles

Several types of dynamic obstacles are considered for the test cases. Figure 3 shows the dynamic obstacle like pedestrian suddenly crossing the road at various speeds.

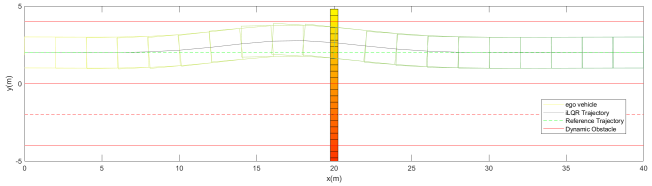
The sequence of boxes of yellow to red color shows represents the pedestrian (gradient shows the movement at



(a) Pedestrian moving at 1 m/s



(b) Pedestrian moving at 2 m/s



(c) Pedestrian moving at 4 m/s

Fig. 3: Dynamic obstacle like pedestrian moving in the lateral direction at different velocities

each time step) and the sequence of boxes of yellow to green color shows the ego vehicle smoothly avoiding the obstacle.

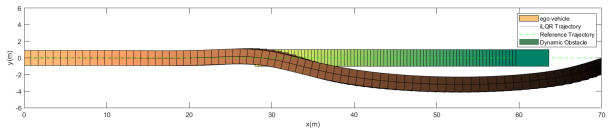


Fig. 4: Obstacle vehicle cutting into the lane of ego vehicle

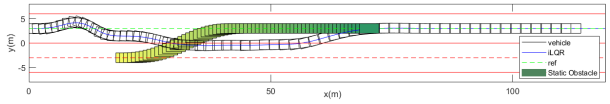


Fig. 5: Obstacle vehicle cutting into the lane of ego vehicle

Figure 4 shows the ego vehicle overtaking a slow moving vehicle ahead of the ego vehicle. The sequence of boxes of yellow to green color shows represents the obstacle vehicle (gradient shows the movement at each time step) and the sequence of boxes of brown color shows the ego vehicle's trajectory for overtaking task.

Figure 5 shows another vehicle suddenly cutting into the ego vehicle's lane. The sequence of boxes of color yellow to green shows represents the obstacle vehicle (gradient shows the movement at each time step) and the sequence of boxes of black color shows the ego vehicle smoothly avoiding the obstacle.

V. CONCLUSIONS

In this paper, We have shown that the Iterative LQR is capable of motion planning for urban areas including evasive maneuvers and emergency situations. Moreover, We have shown the effects of the cost weights on the type of maneuvers autonomous vehicles need to make.

As a part of future work, tuning the cost weights efficiently or having adaptive weights can be considered. Also, the dynamic model of the vehicle needs to be considered for better implementation at high speeds, but solving for dynamic model needs efficient solvers for real-time implementation which has further research scope.

ACKNOWLEDGMENT

The authors would like to thank the Federation of Indian Chambers of Commerce and Industry (FICCI) 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] Lino Figueiredo, Isabel Jesus, JA Tenreiro Machado, Jose Rui Ferreira, and JL Martins De Carvalho. Towards the development of intelligent transportation systems. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*, pages 1206–1211. IEEE, 2001.

[2] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2015.

[3] John K Subosits and J Christian Gerdes. From the racetrack to the road: Real-time trajectory replanning for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 4(2):309–320, 2019.

[4] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for berthaa local, continuous method. In *2014 IEEE intelligent vehicles symposium proceedings*, pages 450–457. IEEE, 2014.

[5] Wenda Xu, Junqing Wei, John M Dolan, Huijing Zhao, and Hongbin Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *2012 IEEE International Conference on Robotics and Automation*, pages 2061–2067. IEEE, 2012.

[6] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *ASME 2010 dynamic systems and control conference*, pages 265–272. American Society of Mechanical Engineers, 2010.

[7] Aliasghar Arab, Kaiyan Yu, Jingang Yi, and Dezhen Song. Motion planning for aggressive autonomous vehicle maneuvers. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 221–226. IEEE, 2016.

[8] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, 2016.

[9] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, Jahan Asgari, and Davor Hrovat. Mpc-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2):265–291, 2005.

[10] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

[11] Jur van den Berg. Extended lqr: Locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost. In *Robotics Research*, pages 39–56. Springer, 2016.

[12] Jur van den Berg. Iterated lqr smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost. In *2014 American Control Conference*, pages 1912–1918. IEEE, 2014.

[13] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[14] Jianyu Chen, Wei Zhan, and Masayoshi Tomizuka. Constrained iterative lqr for on-road autonomous driving motion planning. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE, 2017.

Maximizing Agent Utility in Human-Robot Collaborative Assembly Tasks

Kenneth Shaw, Jaskaran Grover, Changliu Liu

***Abstract*—Human-robot collaboration (HRC) is increasingly important in assembly line tasks as humans and robots have varied strengths that can cooperate safely together to increase efficiency. However, due to the time-varying nature of human collaborators, it is challenging for the robot to efficiently identify the ideal response to the human plan. Currently, most solutions use the trajectory of the human arm as the primary detection method for the human's underlying plan and respond with one appropriate robot action to maximize the utility given the situation. Neither the human nor the robot is expected to provide further help if the task is proving difficult for the other agent, therefore not maximizing the utility of each agent. In this paper, we will improve upon this by grouping robot tasks with the same underlying goal together, allowing the robot to switch between them to maximize the utility of each agent.**

I. Introduction

In factories and many other industrial settings, robots have proven to be efficient at many repetitive tasks. That being said there are also many cases such as small assembly where robots aren't capable of completing the task as effectively. Human-robot collaboration has shown much promise in areas such as manufacturing and has caught the continued interest of academia and industry alike to fill this gap. In a collaborative human-robot environment, physical safety is combined with software checks to ensure comprehensive, guaranteed safety. Software also facilitates the tasks allocated to each agent, robot, and human. This combination can allow robots and humans to work in close proximity together to complete the task, using their complementary strengths most efficiently. This paper examines the allocation strategy of these tasks and proposes one such method.

This allocation strategy must find the proper robot action to match that of the human's that maximizes the utility of the human and the robot. For example, if one were placing a window seal on a car with an assistive robot, there are many different actions the robot and the human could take with various success rates, durations and strain on the human. The robot could attempt to do it fully autonomously but possibly run into issues of alignment and placement. The human could assist the robot through the path, helping it place the door seal. Additionally, the human could complete the whole action by itself. This places a large strain on the human but has a higher guaranteed success rate. There are existing various strategies for this task allocation problem that will be explained below.

II. Related Work

Much work has been previously done on the task allocation problem with a variety of applications. In human-robot collaboration, one common way is by using advanced planners to decide which task to match to the human based on the status of the objects and obstacles in the environment. One such implementation is SHARY, which uses a hierarchy of shared agent tasks and communication mechanisms between the two agents. It uses a human aware task planner that incorporates the feasibility of the task and human load to decide on the correct task to assign. It also maintains a list of suspended and currently active tasks to more efficiently manage the next upcoming tasks for the human-robot team [4].

Additionally, the Berkeley MSC lab has specified human-robot tasks as part of a hierarchy of sub-tasks that can be completed sequentially together to fulfill one larger task. This hierarchy enables the robot to better predict the next action and trajectory of the human which therefore enables the robot to predict and respond earlier to the status of the task [3].

Theory of Mind paradigms use cues from the human, environment, and goals to better estimate the mental state of the human [11]. For instance, eye gaze has already shown promise in finding a target object that a person is looking at [2]. Additionally, inverse planning looks to understand the human's goals based on their abilities in perception, in this case using a supervisor to keep track of the human's understanding and facilitates communication between the agents [5].

In these previously explained human-robot collaborative environments, the robot is responsive to the action of the human and meets it with a singular action that is programmed offline to help the human achieve the underlying intention, where the intention is the underlying goal that would be achieved by conducting such a task. However, when examining human-human teams, this is largely not true. For instance, when human-human teams work together on intentions such as lifting heavy objects, each human monitors the success of the other one and adapts to help and improve the success of the whole team. When playing a game such as soccer, players are constantly monitoring the success of others to improve their own actions and the success of the team. The focus of this paper will be to model this in human-robot collaboration.

In this new human-robot collaborative environment, when a human is struggling on a task the robot will perform further actions to help assist the human. This can better efficiently utilize the resources of the robot as well as the human. In this case, there is an intention or underlying mission that the human must complete. One intention can then have many different tasks that will complete this same end result. The human will provide the initial intention with its first task, but then the robot will decide continued shared tasks to minimize the expected completion time of the intention and maximize the resources available to it.

To ensure safety and allow for trajectory and task functionality, SERoCS (Safe and Efficient Robot Collaboration System) will be used as the underlying basis for the experiments. The robust cognition algorithms monitor the environment and track the human using a Kinect camera by using joint positions and then infer a plan of the human. Once that is understood, an efficient task planning algorithm based on the CFS algorithm dictates the robot plan the long term trajectory. Finally, a safe motion planning and control algorithm for safe human-robot interaction ensures that the long term goal and safety constraints are met. This framework addresses the broad challenges in a human-robot collaborative environment [7].

The structure of the rest of the paper is as follows. In Section III, the mathematical framework of robot task allocation is proposed. In Section IV, the SERoCS framework modifications are discussed. Section V concludes the paper.

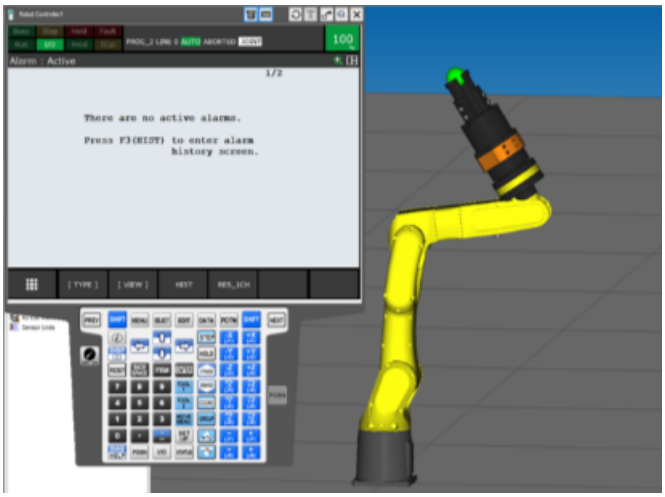


Fig. 1: RoboGuide FANUC Robot Simulator

III. Problem Framework

In a typical human-robot collaborative environment, the robot's task allocation strategy must decide which task to conduct to help the human. This task should attempt to maximize the utility, or usefulness of each agent at completing the task. In this paper, the task that the human is trying to complete is tied closely with the trajectory or action that the human is conducting. Each task, however, will now have an underlying intention. This intention is the desired goal state of the target objects at the end of the successful completion of the tasks. The robot will try to switch tasks to complete the intention faster.

Each intention will have a group of synonymous tasks that can be conducted to achieve the same end intention. As an example, placing a car seal onto a door can be done in three ways, (1) by placing the seal completely automatically with the robot, (2) by having the human help align the robot and door together to complete the task or (3) completely by human action alone. Each of these n tasks within the intention will be defined as a tuple $< P_n, T_n, K_n >$ where P is the probability of success, T is the task time completion distribution, and K is an empirical understanding of the strain and unpleasantness on the human of the action and the process of switching actions. In the same example as above, the completely autonomous procedure might have a fast completion time and little strain on the human, but the success rate could be low. The completely human operated procedure would have a greater amount of human strain but also a higher completion rate and consistent completion time.

Focusing on the task completion time aspect, independent trials can be conducted to gather data and understand this random variable. From prior industrial engineering research, task completion time of tasks such as factory assembly tasks can typically be modelled and fitted to a Weibull distribution [9]. The Weibull distribution is a right skewed distribution, where most tasks will likely be completed earlier rather than later and is defined as follows where $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter of the probability density function:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Once the completion time is fit to this distribution, the distribution's cumulative distribution function is useful to find the area under the curve or the probability of completing the task within a certain amount of time. The CDF is as follows[6]:

$$1 - e^{-(x/\lambda)^k} \quad x \geq 0$$

However, the completion time is only one part of the characteristics that define the cost of the task. Each task has a total success rate over a long period of time as well as a cost of the human strain of conducting and switching to the task. Therefore the total cost can be defined as follows, where there are l task pairs, p_n is the probability of reaching this point, x_n is the duration, and c_n is the cost function of switching tasks.

$$\min : \sum_{n=0}^{l} p_n * x_n * c_n \tag{1}$$

The probability of reaching this point or p_n is the sum of the probability of not completing the previous trials.

$$p_n = 1 - \sum_{m=0}^{n-1} (1 - e^{-(x_m/\lambda_m)^{k_m}}) \tag{2}$$

Combining these we obtain:

$$\min : \sum_{n=0}^{l} (1 - \sum_{m=0}^{n-1} (1 - e^{-(x_m/\lambda_m)^{k_m}}) * x_n * c_n) \tag{3}$$

Therefore, the goal of the robot will be to minimize intention cost or the above function which consequently minimizes the strain on the human as well as task completion time but also guaranteeing task completion, assuming the final $p_l = 0$. It will do this by finding the correct time to switch tasks, or finding the proper vector of x_n to pursue the tasks for. Intuitively, once the success rate of the current task becomes low over time (e.g., the robot is trying to put a window seal on a car door and is failing) then the actions of the robot will switch to a task that will be more likely to be completed.

To organize the relationship between these task tuples, a sequence structure can be proposed. Continuing on the previous example, it is likely that the robot should try to completely autonomously add the door seal to the car first, but then when time dictates it's failing and getting stuck, it should then increasingly allow the human to intervene and complete the action together. Like in this example, the general structure of the net can be understood solely by domain knowledge of the task at hand. Many different permutations can be confidently ignored, and a general sequential structure can be easily presented. A convenient way to represent this is as a Petri Net as below.

Each tasks' forward progress within them are not considered online. For example, if the robot was halfway across the door, that information won't be considered. Only time elapsed on the task will be used as information to minimize the expected cost based on the offline model.

In summary, once the switching times x_n are calculated offline, these will be stored by the robot and can be used at each intention to minimize the estimated cost to complete it. The robot will create an offline compiled list that will dictate which task the robot should complete and when to minimize the total cost of completing the goal or intention, therefore maximizing the utility of the agent team.

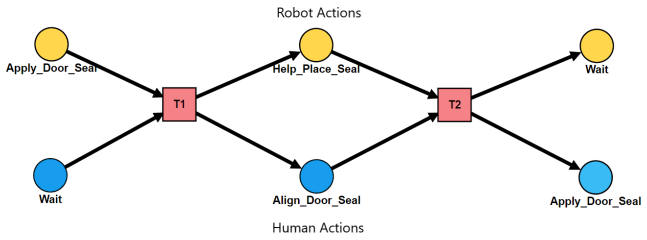


Fig. 2: Petri Net showing an example intention and task sequence, where the robot actions are in yellow, the transitions are in red, and the human actions are in blue.

IV. Experiment Setup

As mentioned previously, SERoCS provided the basic framework for the human-robot collaboration paradigm. Importantly, it provides the ability to set and carry out cartesian goals which are converted into joint based long term trajectories using the CFS algorithm. It uses Kinect environment data and a short term safety planner to keep the robot and human safe together [7]. When SERoCS was used at Berkeley, the MSC lab used FANUC robots along with two computers. To communicate with the robot, a FANUC system called DSA was used, where the desired motions were converted into torque commands and provided to the robot using EtherCat. One host computer ran the longer term plan and trajectory optimization, where a second target computer ran the safety controller as well as communicated with the robots. At CMU, even if the one physical robot we have is similar, the control mechanism for the robot is different and only one PC was desired. In this case, a UDP protocol called StreamMotion is used to command angle positions for each of the six

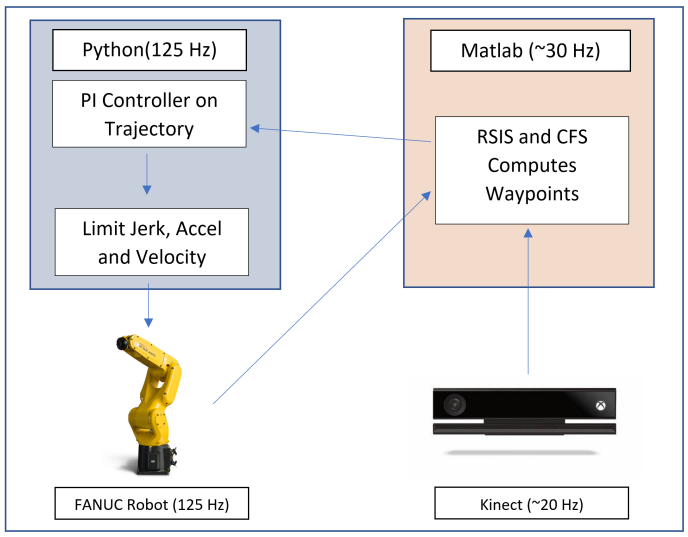


Fig. 3: Final Matlab to StreamMotion Trajectory Framework

joints of the FANUC robot or the accompanying simulator. Within every 8ms, the computer must respond with a target trajectory before the next status packet is received from the robot, which consists of the position and other diagnostic data. This immediately posed significant problems to the implementation because the long term planner can only be solved at a frequency of 30Hz.

To solve this problem, first, an experimental interface and a decoder for StreamMotion packets was created with Matlab and used to communicate with the FANUC robot. The intention was to integrate this with SERoCS since it was also written in Matlab and Simulink. After much experimentation, however, it continued to result in a MOTN-604 error because Matlab was not expedient enough to satisfy the 8ms constraint consistently. To avoid this issue, a Python-based UDP library was trialed. Thankfully, the Python provided bytes object was very useful for decoding and prototyping could be done very quickly. By itself, Python could easily keep up with the StreamMotion packets from the robot simulator. However, Python must now also communicate with the Matlab code running at approximately 30Hz. To architect this, a Matlab wrapper for the Python terminal was used to communicate positions in joint space to Python. To allow for the StreamMotion packets to be sent out possibly simultaneously and allow for Matlab to open the Python terminal when desired, the StreamMotion portion of the Python code was placed in its own high priority thread. Initially, the Matlab data was simply cached and repeated to the robot, effectively a zero hold method.

It was then found that the trajectory plan must be smoothed online to follow manufacturer-defined jerk, acceleration and velocity constraints before sending them to the robot. First, it was thought that a trajectory generation method like polynomials splines or lines could be used to smooth out the trajectory and avoid those constraints [1] [8]. To maintain expedient calculations, a line generation method was tried between the current point and next target point at each time the data was received from Matlab, where the expected amount of time

between each Matlab command was used to determine the number of points to generate. If a point from Matlab was received earlier than expected, the line would be redrawn to the new target position. If it was received later, then the robot would enter a zero order hold until the point was received. However, the constraint on jerk was still exceeded very often. Therefore, to meet this restriction, it was decided to attempt to apply these constraints online at 125 Hz and constrain the trajectory before sending it to the robot. The velocity, acceleration, and jerk was calculated as follows, where δt is the constant step size between two consecutive time intervals i.e. $\delta t = t_k - t_{k-1}$

$$v(t_k) = \frac{q(t_k) - q(t_{k-1})}{\delta t},$$
$$a(t_k) = \frac{v(t_k) - v(t_{k-1})}{\delta t} = \frac{q(t_k) - 2q(t_{k-1}) + q(t_{k-2})}{\delta t^2},$$
$$j(t_k) = \frac{a(t_k) - a(t_{k-1})}{\delta t} = \frac{q(t_k) - 3q(t_{k-1}) + 3q(t_{k-2}) - q(t_{k-3})}{\delta t^3}$$

and these constraints were applied:

$$v_{min} \leq v(t) \leq v_{max},$$
$$a_{min} \leq a(t) \leq a_{max},$$
$$j_{min} \leq j(t) \leq j_{max} \ \forall t$$

where v_{min}, v_{max} are the minimum and maximum limits on the velocity allowed by the robot manufacturer and acceleration and jerk are a and j respectively.

For instance, the application of the constraints was done as below for jerk:

$$a(t)_{max} = j_{max} + a(t-1),$$
$$v(t)_{max} = a(t)_{max} + v(t-1),$$
$$d(t+1) = d(t) + v(t)_{max}$$

for acceleration:

$$v(t)_{\max} = a_{\max} + v(t-1),$$
$$d(t+1) = d(t) + v(t)_{\max}$$

and for the velocity:

$$d(t+1) = d(t) + v_{\max}$$

However, this proved to be unstable. For example, in a ramp input, the actual position would initially be much below the desired position. The robot will maximize the jerk and acceleration constraints to meet the desired positional trajectory. Once meeting the trajectory, the velocity and acceleration would be much larger than that of the ramp input. The robot would then use a maximum negative jerk, attempting to correct itself. However, then the velocity and acceleration would be much lower than the ramp input. This process would repeat throughout the trajectory and become unstable under larger inputs.

To improve the stability, a PI controller was tried to facilitate the tracking of the robot to the desired trajectory. This promised to easily smooth out the trajectory, be stable and minimize error while being fast enough for online use. In this case, let $q_d(t)$ be the desired angle at the current timestep, and $q_d(t-1)$ be the previously desired angle. The error is defined as $e(t) = q_d(t) - q_d(t-1)$. A feedback and feedforward controller where PI is the feedback component was then used as follows:

$$q_d(t) = q_d(t-1) + K_p e(t) + K_i \int_0^t e(t)dt \qquad (4)$$

As demonstrated by the plot attached, this fulfilled the requirements for the robot trajectory.

This work completed on the FANUC robot will be integral to future work on the robot in the lab for human-robot collaboration and other experiments. Further research in task assignment will be conducted on the robot in the future.

V. Conclusion

The work completed on the FANUC robot as explained in the previous section will be very helpful for future projects in the lab. We have also theoretically shown that this task assignment paradigm has the ability to more efficiently optimize a human-robot collaborative assembly task and maximize the utility of each agent. However, in the future, additional work must be done on experimental situations. This will enable a better understanding of various task types and their cost and duration distributions. As for other research directions, this basic assignment structure could use additional data to make decisions on what the robot needs to accomplish, including verbal communication, eye gaze and other Theory of Mind type body cues. One could evaluate the skill and fatigue level of the human and adjust the degree of robot collaboration and automation in this case extending to different task action pairs online [10]. The robot could also provide additional information to help the human orient in the task. Finally, there was an assumption that forward progress to one task will be ignored when deciding whether to move to the next one. This was chosen to simplify the calculations of the issue but either a more advanced model or reinforcement learning could be considered to choose the optimal time to switch between tasks. Again, the framework designed here in the experiment section will be useful with SERoCS and many other experiments in the lab for the future.

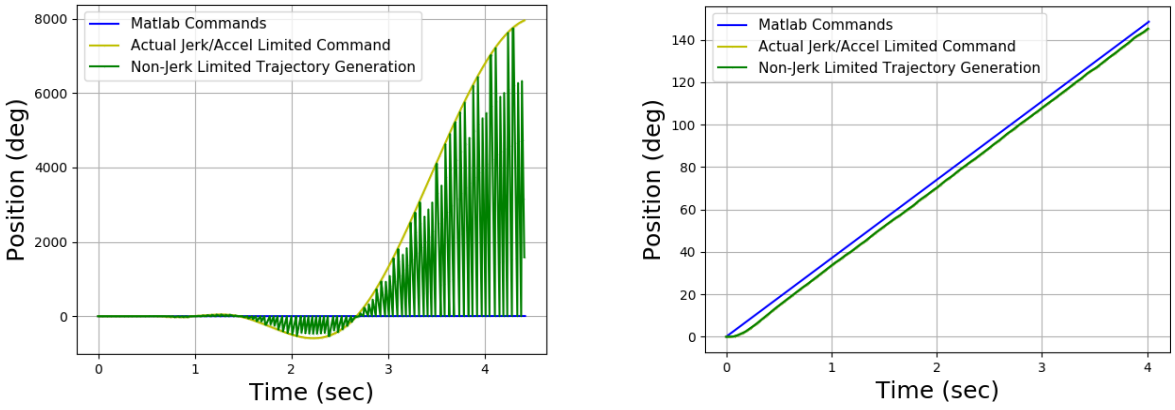
VI. Acknowledgements

This work was supported by the Robotics Institute Summer Scholars Program, the National Science Foundation REU program, and related to National Science Foundation Award #1734109. The authors would additionally like to recognize FANUC for their technical support.

References

[1] P. Allen, "Coms4733: Trajectory planning, class notes," November 2015.
[2] R. M. Aronson and H. Admoni, "Semantic gaze labeling for human-robot shared manipulation," in *ACM Symposium on Eye Tracking Research and Applications (ETRA)*. ACM, June 2019.
[3] Y. Cheng, L. Sun, and M. Tomizuka, "Towards better human robot collaboration with robust plan recognition and trajectory prediction," *arXiv preprint arXiv:1903.02199*, 2019.

[4] A. Clodic, H. Cao, S. Alili, V. Montreuil, R. Alami, and R. Chatila, "Shary: a supervision system adapted to human-robot interaction," in *Experimental robotics*. Springer, 2009, pp. 229–238.
[5] S. Devin and R. Alami, "An implemented theory of mind to improve human-robot shared plans execution," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 319–326.
[6] S. Kotz, N. Balakrishnan, and N. L. Johnson, *Continuous multivariate distributions*. Wiley, 2000.
[7] C. Liu, T. Tang, H.-C. Lin, Y. Cheng, and M. Tomizuka, "Serocs: Safe and efficient robot collaborative systems for next generation intelligent industrial co-robots," *arXiv preprint arXiv:1809.08215*, 2018.
[8] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, 2003.
[9] B. Rummel, "Beyond average: Weibull analysis of task completion times," *Journal of Usability Studies*, vol. 12, no. 2, pp. 56–72, 2017.
[10] B. Sadrfaridpour, H. Saeidi, J. Burke, K. Madathil, and Y. Wang, "Modeling and control of trust in human-robot collaborative manufacturing," in *Robust Intelligence and Trust in Autonomous Systems*. Springer, 2016, pp. 115–141.
[11] B. Scassellati, "Theory of mind for a humanoid robot," *Autonomous Robots*, vol. 12, no. 1, pp. 13–24, 2002.



(a) Linear Interpolation (Unstable)

(b) PI Tracking Controller (Stable)

Fig. 4: One joint trajectory under ramp input

Degeneracy Detection for RGB-D Odometry

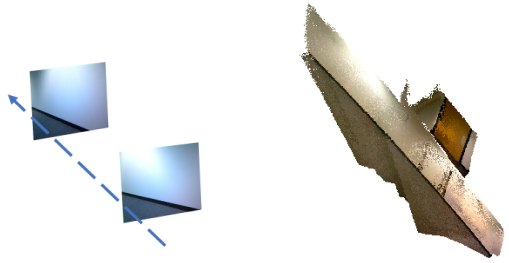
Zilin Si[1], Ming Hsiao[2] and Michael Kaess[3]

***Abstract*— In this work, we propose an online degeneracy detection algorithm for RGB-D odometry estimation. In the environments with insufficient visual textures or geometric structures, most localization systems may find it difficult to estimate accurate motions due to the lack of information for full 6 DoF pose constraint. To achieve better robustness for state estimation, is it crucial to detect and handle these degeneracy situations. Our method is developed basing on a real-time CPU-based fast RGB-D odometry method [1] that uses both photometric and geometric information. Within the underlying nonlinear optimization process, our solution determines the well-constrained and degenerated dimensions based on dynamic thresholds for both the photometric and geometric parts, and combine them together to find the degenerate directions of the overall system. We evaluate our approach using both online datasets and self-recorded datasets and demonstrate its ability of detecting degenerate situations accurately.**

I. INTRODUCTION

Visual odometry has been studied for a long time in the SLAM community. Cameras is an essential and useful sensor for SLAM systems, which can provide high resolution visual information. With sufficient textures detected by camera, feature-based visual odometry has been developed to offer accurate localization, such as monocular visual odometry [2] and ORB SLAM [3]. On the other hand, range sensors can be used to estimate motion, such as applying ICP algorithm [4] directly. Using RGB-D cameras which combines vision and range sensing in SLAM system becomes commonplace in recent years. By expanding information captured by RGB-D camera, algorithms have been designed for RGB-D odometry which can provide more accurate results. For instance, beyond the indirect methods which mostly refer to sparse feature-based odometry [5], dense RGB-D odometry has been developed by minimizing the photometric error of each pixel [6]. This method incorporates each pixel's intensity as well as depth information into optimization, which can provide more accurate camera ego-motion estimation.

Most RGB-D odometry methods can perform well in environments with adequate visual features and geometric structures. However, imperfect conditions of the environment can result in bad localization. Degeneracy is one of the most common problem that plagues reliable localization. Generally, the degeneracy can be considered as losing partial constraints for estimation. For instance, in Figure 1, if we walk along the corridor and holding the camera which only captures the wall and floor, it is hard to know whether the camera is moving forward, backward, or staying static only from the input sequence of images. So in this case, the translation direction along the corridor is degenerated. Basically, there are two common ways to handle degeneracy situations: One is to offer more constraints by providing a multi-sensor system like visual inertial odometry [7], in which the measurements from an inertial measurement unit (IMU) can compensate the lack of visual information. The other is actively detecting and handling degeneracy to make more reliable prediction and update. [8] proposed a method to handle environmental degeneracy, in which they analyze and separate well-constrained directions from degenerate directions. This method can be used for the general optimization-based system.



(a) Trajectory and captured image (b) Mapping from degeneracy case

Fig. 1. This is an example of degeneracy. If the camera only captures the wall and floor as (a), for two frames, no matter we slide along the translation direction parallel to two planes or not, we can always align them. Thus this direction is degenerate. As a result, the trajectory of motion is not accurate and (b) shows the overlap of wall and door because of the incorrect motion estimation.

In this paper, we actively analyze and detect degeneracy situations in an online localization system. Our proposed degeneracy detection method for RGB-D odometry is based on the fast RGB-D odometry proposed in [1], which already combines a geometric-based method called *iterative projected planes* and two photometric-based methods to estimate the odometry of the sensor for better robustness. If only one part has degenerate directions, the other part can compensate for it directly under its joint optimization framework. However, if both parts run into degeneracy, the algorithm cannot estimate the odometry accurately. Therefore, our proposed method detects the degeneracy separately in each part, and finds the best constraints for both of them together. Only when both parts degenerate in the same directions, we regard them as the degenerate directions of the overall system.

[1]Zilin Si is with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China sizl@shanghaitech.edu.cn

[2]Ming Hsiao is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA mhsiao@andrew.cmu.edu

[3]Michael Kaess is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA kaess@cmu.edu

Our contributions in this paper are as follows:

- Detecting the degeneracy on both parts (geometric and photometric) online.
- Combining two parts together and find the common degenerate directions.

This paper is organized as follows. The next section discusses related work. The basic RGB-D odometry system is introduced in Section III. We present and analyze the approach of degeneracy detection in Section IV. Experiments and results are provided in Section V. Finally, we draw the conclusion and discuss future work in Section VI.

II. RELATED WORK

Our degeneracy detection algorithm works on RGB-D odometry which is well studied in two different ways. RGB-D odometry can be realized with feature-based method, such as [5] presented a fast RGB-D system that reached real-time frequency by aligning sparse features. Features can typically represent the image, and due to the reasonable calculation workload, the system can achieve real-time localization efficiently. On the other hand, color information with depth information makes the direct motion estimation method more robust. ICP [9] is widely used in SLAM, including RGB-D camera implementation. [10] proposed semi-dense visual odometry for RGB-D camera which registers 2D semi-dense region with 3D semi-dense map by implementing approximate nearest neighbor fields. As [11] presented the energy-based approach to RGB-D odometry, they estimated the motion by minimizing the photometric errors in a coarse-to-fine manner.

Being Robust to various environments including but not limited to degeneracy is essential to RGB-D odometry. To make odometry robust in a wider environment, [12] bases on directly minimizing photometric error method, and provides the robust error function to reduce the influence of noise and outliers. Also, it can be realized in real-time with only CPU computational source. [13] proposed a robust RGB-D SLAM algorithm by heuristically switching between RGB-BA and RGB-D-BA to handle the inaccurate estimation from depth information in poor geometric structure environment or other failure modes. This algorithm makes the system perform well in a general environment. In [1], authors developed a keyframe-based dense planar SLAM (KDP-SLAM) system. This system provided a fast dense RGB-D odometry which combines the geometric estimation with photometric estimation. By minimizing the residual of the energy function, motion can be recovered more accurately due to incorporating the measurement of geometric structure and photo intensity alignment.

Under the poor geometric structure and insufficient texture environment, it is hard to align two frames regardless of using either feature-based method or direct method. Degeneracy is a kind of poor situations which affect the accuracy of motion estimation profoundly. To handle this issue, [8] proposed an online approach to analyze and separate the well-constrained and degenerate directions in optimization-based problems. Even though some directions are under degeneracy, there are still other directions that can provide correct estimation and be used to update the state of space. We adopt this idea in our work to divide all directions to well-conditioned and degenerate directions. Moreover, [14] proposed a degeneracy detection method for plane feature-based measurement and then use inertial measurement unit (IMU)'s information to compensate for the lack of constraints. The experiments were conducted on an RGB-D-inertial integrated system. In our geometric part, we use planar-based method to detect the number of degenerate directions.

III. PRELIMINARIES

As the foundation of our degeneracy detection algorithm, we work on fast dense odometry [1] mentioned in Section II. Since RGB-D camera can offer depth information with scale as well as intensity information, we can optimize the problem from both geometric and photometric views to improve the robustness. Similar to ICP algorithm, the geometric optimization is achieved with iterative projected plane (IPP) approach. And we minimize the photometric errors to align two frames as photomeric optimization, which is similar to [11]. The total energy function can be represented as follows:

$$E_{total} = E_{geo} + \lambda E_{pho} \tag{1}$$

where λ is used to adjust the relative weighting of two terms in the optimization.

This system balances between computational efficiency and accuracy. To achieve this goal, we set two kinds of frames: *reference frame* and *general frame*. All the general frames' transformation is related to the nearest reference frame, and only rough estimation is provided to save calculation source. The rough estimation is achieved by IPP and dense RGB-D odometry in a pyramid way. We set the output of IPP to the input of the first level of pyramid dense RGB-D odometry. Also, each level's output is the initialization of the next level.

A new reference frame R_j is decided when its location is far from the last reference frame R_{j-1}. And the reference frame's motion has to be estimated with a precise transformation to make sure the accuracy of the whole system. The precise estimation is implemented by IPP and semi-dense RGB-D odometry. For the semi-dense part, it is mostly in the same way as rough estimation, but with one more semi-dense level at the bottom of pyramid as Figure 2. Each frame's pose is initialized with the last frame's pose to speed up the convergence of optimization.

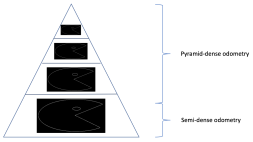


Fig. 2. Pyramid/Semi-dense structure. In our course-to-fine manner, the upper three levels are used for pyramid dense method to estimate rough motion. The bottom level has the original size of image, so we cannot realize real-time calculation by using dense method. Therefore, we apply semi-dense method to the bottom level. Based on rough motion estimation, the precise motion estimate is recovered by adding semi-dense approach.

The next part will introduce IPP, pyramid dense RGB-D odometry and semi-dense odometry individually.

A. Iterative Projected Plane(IPP)

IPP algorithm is similar to ICP point-to-plane method, but it bases on iteratively energy minimization to find the association between two frames' sub planes, which are the small planar patches extracted from the depth image. Assume now we want to associate the reference frame R_r and current frame F_n. From the raw depth data, it is hard to extract accurate planar patches due to the noise, so we smooth the planar areas of image with low-intensity gradients at the beginning. After the preprocessing, a set of sub planes $\{p_i^F\}$ and $\{p_j^R\}$ can be extracted by finding patches in which majority of points' normal vectors have the uniform direction. Each plane p_i can be represented by the center point $c_i = [x_i,\, y_i,\, z_i]^\top$ and normal vector $n_i = [a_i,\, b_i,\, c_i]^\top$. Given the current prediction of transformation, we can project each sub plane p_i^F of current frame F_n to reference frame R_r. The sub plane p_i^F is associated with p_j^R if the center point c_i^F can be projected onto the p_j^R. In each iteration, the energy function with m couples of sub planes found is:

$$\begin{aligned} E_{geo} &= \sum_{i=1}^{m} ||(R(\xi)c_i^F + t(\xi))^\top n_i^R + d_i^R||^2 \\ &\approx \sum_{i=1}^{m} ||A_i\xi - e_i||^2 \end{aligned} \tag{2}$$

where $\xi = [\beta,\, \gamma,\, \alpha,\, t_x,\, t_y,\, t_z]^\top$ is the optimization object of three rotations$(\beta,\, \gamma,\, \alpha)$ and three translations$(t_x,\, t_y,\, t_z)$; $R(\xi)$ and $t(\xi)$ are the corresponding rotation matrix and translation vector of ξ; d_i^R is the distance parameter of the plane p_i^R:

$$a_i^R x + b_i^R y + c_i^R z + d_i^R = 0 \tag{3}$$

Then the update of state vector is:

$$\Delta\xi = (A^\top A)^{-1} A^\top e \tag{4}$$

where A is the stack of all A_i.

IPP algorithm can only perform well in the environment with sufficient and unique sub planes matched. If in the complex surrounding without enough planes or in contrast, in the environment with simple structures where most of the sub planes are indistinguishable, the algorithm cannot provide accurate estimation. Therefore we have to incorporate photometric optimization to improve the accuracy and robustness of algorithm.

B. Pyramid Dense RGB-D Odometry

The pyramid dense RGB-D odometry method mostly follows [11] and [12] by minimizing the photometric error in a coarse-to-fine manner except that here we utilize the Laplacian of the images instead of grayscale images to mitigate the assumption of brightness consistency. We optimize the energy function from the coarsest level, and each time we initial the next level with the output of the last level to accelerate computation. In each iteration, with the known temporarily-estimated motion ξ, the photometric error can be computed as:

$$\begin{aligned} E_{pho} &= \sum_{i=1}^{n} ||I_F(x_i) - I_R(w(\xi, x_i)||^2 \\ &\approx \sum_{i=1}^{n} ||J_i\xi - r_i||^2 \end{aligned} \tag{5}$$

where I_F and I_R are the Laplacian images of current frame F_n and reference frame R_r; w is the warping function that project a valid pixel x_i from F_n given the motion ξ to reference frame; J_i is the Jacobian of the pixel x_i in the downsampled Laplacian image of R_r with respect to a 6 DoF perturbation, and r_i is the residual between each valid pixel pair of x_i in R_r and projected pixel from F_n; n is the number of valid pixel pairs that are used in each iteration. Then we can obtain the updating of ξ:

$$\Delta\xi = (J^\top J)^{-1} J^\top r \tag{6}$$

where J is the stack of all J_i.

C. Semi-dense RGB-D Odometry

The semi-dense RGB-D odometry is the bottom level of the pyramid dense RGB-D odometry which uses the information of the entire image. However, the huge number of pixels' calculations can not be achieved fast on CPU. To reduce the burden of calculation, we only use pixel with high-intensity gradients to typically represent the image. One issue is that usually pixels with high gradients are located at the discontinuous in 3D world frame, the depth measurements of these places are often inaccurate. Thereby, we pick the pixels with better depth measurements nearby these pixels to form the energy function. Then apply the same method used for pyramid-dense part to solve the transformation estimation iteratively. The semi-dense method only works on reference frames to refine the motion.

D. Joint Odometry Estimation

No matter for reference frames or general frames, they both use the conjunct method of geometric and photometric parts, and solve the non-linear least-squares optimization problem via Levenberg Marquardt method. The architecture of the whole system is depicted in Figure 3.

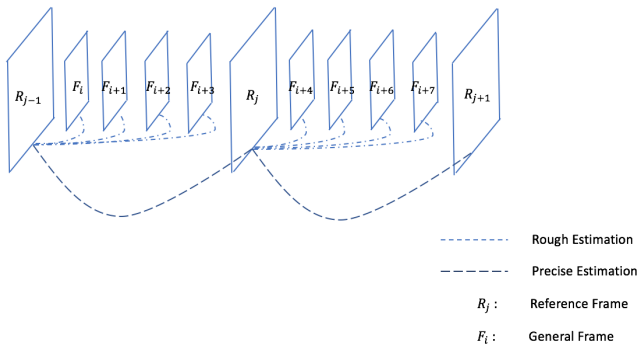


Fig. 3. The overall system. The transformation of each frame is related to the nearest reference frame. Reference frame's motions are estimated precisely, and general frame's motions are estimated roughly.

IV. METHODOLOGY

A. Problem Statement

The energy-based method is widely used in motion estimation. By iteratively linearizing the non-linear problem and minimizing the residuals, the transformation between two frames can be recovered. Generally, we can represent the linearized formula in each iteration as

$$E \approx ||Jx - r||^2 \tag{7}$$

where J is the Jacobian matrix, x is update state vector of six degree of freedom, and r is the residual. Then apply first order optimization to solve it:

$$\begin{aligned} J^\top Jx &= J^\top r \\ x &= (J^\top J)^{-1} J^\top r \end{aligned} \tag{8}$$

To analysis the property of degeneracy, do eigenvalue decomposition to $J^\top J$:

$$\begin{aligned} J^\top J &= V \Lambda V^\top \\ &= [v_1,\, v_2,\, ...,\, v_n] \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda 2 & ... & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & ... & \lambda_n \end{bmatrix} V^\top \end{aligned} \tag{9}$$

where v_i is one of the eigen vectors, and λ_i is the corresponding eigenvalue. Also, here we arrange the eigenvalues λ_1, λ_2, ..., λ_n in increase order, such that $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. By decomposing the x onto eigen vectors, we can represent x as:

$$x = [v_1,\, v_2,\, ...,\, v_n][a_1,\, a_2,\, ...,\, a_n]^\top \tag{10}$$

where v_1, v_2, ..., v_n are the same set of eigen vectors of $J^\top J$, a_1, a_2, ..., a_n are the linear combination coefficients. Thus when multiply $J^\top J$ on x, it works as doing shrink or stretch on each eigen vector direction:

$$\begin{aligned} J^\top Jx &= (V \Lambda V^\top)(V[a_1,\, a_2,\, ...,\, a_n]^\top) \\ &= [v_1,\, v_2,\, ...,\, v_n] \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda 2 & ... & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & ... & \lambda_n \end{bmatrix} I[a_1,\, a_2,\, ...,\, a_n]^\top \\ &= [v_1,\, v_2,\, ...,\, v_n][\lambda_1 a_1,\, \lambda_2 a_2,\, ...,\, \lambda_n a_n]^\top \end{aligned} \tag{11}$$

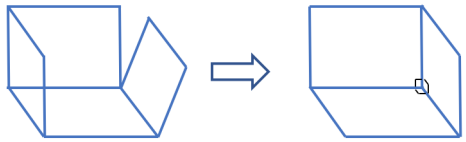
If the problem is solved under the well-constrained environment, $J^\top J$ is fully ranked and invertible since all eigenvalues are positive, which can give us a confident and right solution to estimate state vector x. However, if one or more directions loses the constraint, the corresponding eigenvalues become zero, and the matrix $J^\top J$ is degenerate. We notice that in reality, eigenvalues are not exactly equal to zero but approach to zero due to the noise of data. If assuming there is one degenerate direction, $\lambda_1 \approx 0$, under this situation, a_1 can be an arbitrary value that is not constrained, which leads to an incorrect solution to x.

From the analysis above, the degeneracy situation leads to ill-conditioned matrix $J^\top J$ and affects the accuracy of the solution. However, it is evident that after decomposing the solution onto eigen vectors, only degenerate directions are not well estimated, but other constrained directions are not affected and still can be used to update state vector. Based on that, we can detect the degeneracy directions first, then partially update the transformation only on well-constrained directions.

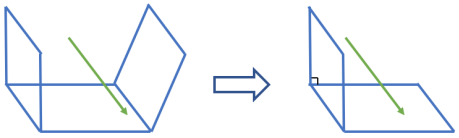
Based on our RGB-D odometry, the degeneracy may arise in diverse directions for the geometric part and photometric part. Since IPP algorithm relies on planar detection, detecting a surrounding which is lack of sufficient geometric structures will lead to degeneracy. However, in the photometric part, degeneracy or not depends on the pattern of pixels with large intensity gradients. If there are not enough patterns to constrain all directions, the pyramid/semi-dense odometry suffers from degeneracy. So degeneracy may occurs in both parts but in different directions. We propose to detect the degenerate directions separately and then combine them to get the smallest number of degeneracy directions.

B. Degeneracy Detection for IPP

As mentioned in Section III A, IPP algorithm works on plane detection and matching. Since we need to estimate the transformation with six degree of freedom, three for rotations and three for translations, at least three sets of planes that are not parallel to each other can constrain all directions. As shown in Figure 4 (c), if all the planes detected are parallel, three degrees are free including one rotation in the normal vector's direction of planes and two translations of the directions parallel to the plane; if planes detected are in two direction groups as Figure 4 (b), one translation degree is free; only when there are three or more direction groups as Figure 4 (a), this problem can be fully constrained. Each plane can be represented by the center point c_i and normal



(a) Fully constrained case



(b) Insufficient constrained case with one free directions

(c) Insufficient constrained case with three free directions

Fig. 4. General constrained/unconstrained cases are shown on the left and the equivalent representation on the right. (a) is fully constrained in all directions; (b) has one free direction (for translation); (c) has three free directions(one for rotation and two for translations).

vector n_i which is perpendicular to the plane. In order to detect the number of degenerate directions, we construct matrix N with all sub planes' normal vector n_i of the current frame F_n:

$$N = [n_1, n_2, ..., n_N] \tag{12}$$

Then conduct eigenvalue decomposition to matrix $M = NN^\top$:

$$M = V\Lambda V^\top = [v_1, v_2, v_3] \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} V^\top \tag{13}$$

where V is the orthonormal matrix whose columns are eigen vectors v_i, and Λ is the diagonal matrix whose diagonal elements are eigenvalues λ_i (i = 1,2,3).

The number of degenerate directions can be decided by the effective rank. For the case in Figure 4 (b), the rank of matrix M should be 2 ideally. However, with the noise of data, the matrix M always has full rank and we have to discriminate the false rank with a threshold. In our experiment, the ratio of the largest and the smallest eigenvalue is a proper threshold which can be achieved online. Therefore, the number of eigenvalues who are less than threshold can indicate the number of degenerate directions. The cases in Figure 4 (a), (b), (c) correspond to 0, 1, 2 eigenvalues that are less than threshold respectively.

Besides the number of degenerate directions, we also have to know which directions are degenerate. We assume there are m degenerate directions. Back to the IPP algorithm, in each iteration we have the matrix $A^\top A$ in Eq. 4, and conducting eigenvalue decomposition on $A^\top A$ gives us a set of eigenvalues $\{v_i^A\}$. The smallest m eigen vectors represent the degenerate directions.

C. Degeneracy Detection for Pyramid/Semi-Dense RGB-D Odometry

Also, we need to detect degeneracy for the photometric part. Different from the geometric part, the pyramid/semi-dense RGB-D odometry uses intensity alignment to optimize the problem. We have the Jacobian matrix of all selected pixels and try to minimize error function between two frames. Therefore we can apply the method in [8] to the matrix $J^T J$ and analyze eigenvalues to find the number of degeneracy as well as the corresponding directions. In [8], they use a predefined threshold to distinguish degenerate and well-constrained direction groups. In our case, we set the threshold with a dynamic ratio of the maximum eigenvalue and the minimum eigenvalue as same as the threshold of degeneracy detection on IPP algorithm. The number of eigenvalues that are less than threshold is the number of degenerate directions. And the corresponding eigen vectors for these eigenvalues represent the degenerate directions.

D. Combination and Updating

After the geometric and photometric degeneracy detection via planar-based and energy-based methods, we have the number and the directions of degeneracy for both parts respectively. However, they may degenerate in different directions and sometimes compensate each other in some directions. So integration of two parts is necessary to find the common degenerate directions.

The six degrees of freedom can be seen as a $\Re^6$ space which is composed by two perpendicular subspaces. And the subspace spanned by degenerate eigen vectors and the subspace spanned by well-constrained eigen vectors can be seen as these two subspaces. Therefore, the idea is to find the NULL space of the constrained subspaces. One way is to stack all constrained eigen vectors from both parts, as $[v_1^A, v_2^A, ..., v_n^A]$ and $[v_1^J, v_2^J, ..., v_m^J]$, where n and m are the number of well-constrained eigen vectors for geometric part and photometric part. Then conduct eigenvalue decomposition to the matrix $W^\top W$ where

$$W = [v_1^A, ..., v_n^A, v_1^J, ..., v_m^J] \tag{14}$$

If eigenvalue λ_i is close to zero, the corresponding eigen vector v_i^W is one of the overall degenerate directions. This method outputs all degenerate directions for the system. After that, the update of state vector can be projected onto well-constrained directions as

$$\begin{aligned} V &= [v_1^W, ..., v_m^W, v_{m+1}^W, ..., v_n^W]^\top \\ V_f &= [v_1^W, ..., v_m^W, 0, ..., 0]^\top \\ \Delta\xi &= V^{-1} V_f \Delta\xi \end{aligned} \tag{15}$$

where the last $n - m$ eigen vectors are close to zero.

The total degeneracy detection algorithm can be summarized as:

- For geometric part, detect the number of degenerate directions via planar-based method.
- For photometric part, detect the number of degenerate directions via energy-based method.
- Find the total degenerate directions as well as its number.
- Project the update of state vector onto well-constrained directions.

V. EXPERIMENTS AND RESULTS

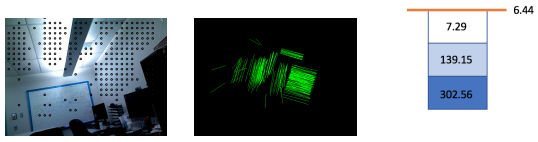
A. Degeneracy of IPP

We have tested the geometric part's degeneracy situation on self-recorded datasets and tum datasets. As shown in Figure 5, 6, 7, three different situations are covered in.

The first one is recorded in the room, and one corner can constrain motion in all dimensions. Figure 5 (b) shows the normal vectors of all extracted planes in Figure 5 (a) marked with dots (center points), which have at least three different directions. Also, eigenvalues of the normal matrix can be seen in Figure 5 (c). All eigenvalues are above the threshold means no degenerate direction.

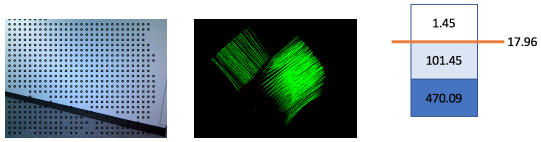
The second dataset is recorded along the corridor, so only one floor and one wall can be seen. As Figure 6 (b) shows, normal vectors can be separated in two groups of directions, and the minimum eigenvalue is below the threshold in Figure 6 (c) which means one direction is degenerate.

The third case comes from the tum dataset, freiburg3_nostructure_notexture_far. It only includes one whiteboard so all the normal vectors are in one direction. Two eigenvalues are below the threshold since three degenerate dimensions.



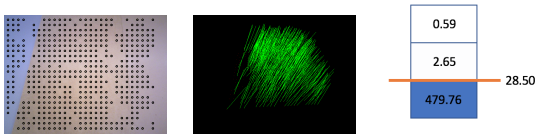
(a) Original image with captured planes (b) Normal vectors (c) Eigen values and threshold

Fig. 5. There are more than three direction groups of planes, so all directions are constrained and all the eigen values are above the threshold.



(a) Original image with captured planes (b) Normal vectors (c) Eigen values and threshold

Fig. 6. There are only two direction groups of planes, so one dimension is degenerate and one eigen value is below the threshold.

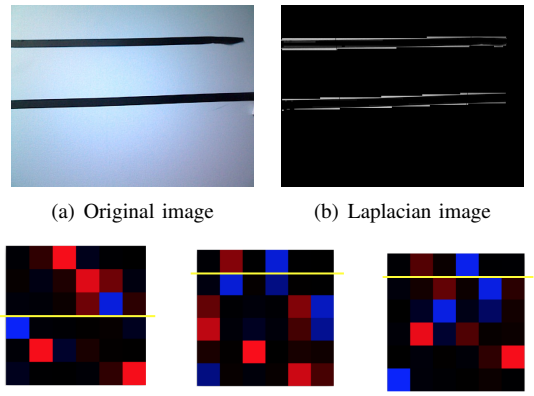


(a) Original image with captured planes (b) Normal vectors (c) Eigen values and threshold

Fig. 7. Only one direction group is found for planes, so three dimensions are unconstrained and two eigen values are below threshold.

B. Degeneracy of Dense/Semi-dense Odometry

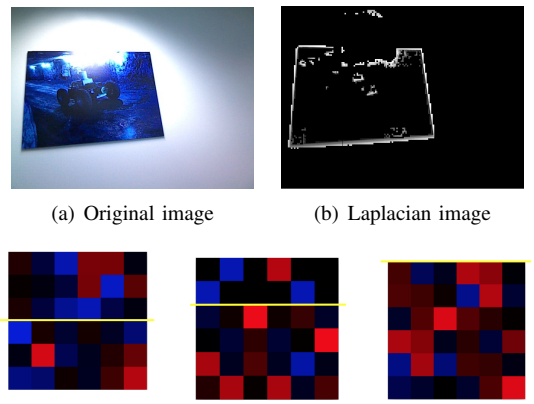
We test the photometric degeneracy detection on several datasets, and depict the typical results in Figure 8 and 9.



(a) Original image (b) Laplacian image

(c) Eigen vectors and threshold for geometric part (d) Eigen vectors and threshold for photometric part (e) Eigen vectors and threshold for overall system

Fig. 8. One direction degeneracy. For the photometric part, the eigen vector in the first row is dominated by the combination of rotation of y-axis and translation of x-axis.



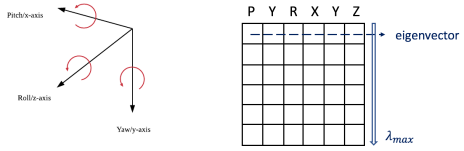
(a) Original image (b) Laplacian image

(c) Eigen vectors and threshold for geometric part (d) Eigen vectors and threshold for photometric part (e) Eigen vectors and threshold for overall system

Fig. 9. Fake degeneracy. For the photometric part, the combination of rotation of y-axis and translation of x-axis as well as the combination of rotation of x-axis and translation of y-axis will lead to fake degeneracy. However, from (e) we can see that this kind of fake degeneracy will be compensated by geometric part.

For Figure 8 (a), there is one white wall with two parallel lines captured and one direction is degenerate as

shown in Figure 8 (d). In the eigen vectors matrix, the red block represents negative value, the blue block represents positive value, and the lighter block means the larger absolute value and vice versa. Since the eigenvalues are arranged in increasing order, the smallest eigenvalue corresponds to the first row. Also the order of elements in one eigen vector is pith, yaw, roll for rotation, t_x, t_y, t_z for translation as Figure 10. So the combination of rotation around y-axis and translation on x-axis dominants the degeneracy. The reason behind is that when current frame slides on x-axis, the two parallel lines can always overlap with the reference frame's pattern. Therefore, it degenerates in translation of x-axis direction. For the rotation around y-axis, since we use numerical differentiation to build Jacobian matrix, when adding a small perturbation on the rotation of y-axis, it performs the same as translating on x-axis.



(a) Coordinate system (b) Matrix of eigen vectors

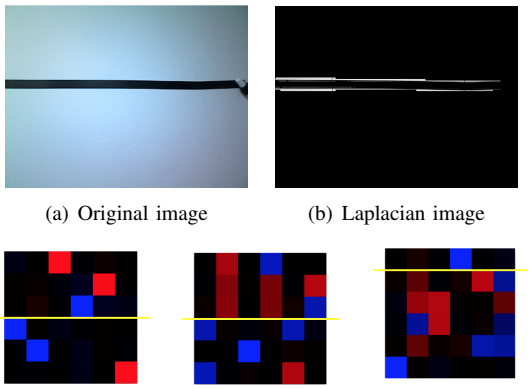
Fig. 10. The coordinate system is on the left. The matrix of eigen vectors is on the right. The order of six direction in one eigen vector is pitch, yaw, roll, x, y, z. And each row represents one eigen vector, all eigen vectors are arranged in increasing order from top to bottom.

As for Figure 9, at the first glance we think it should perform well and constrain all the directions. However, there are two degenerate directions. By looking at the eigen vectors matrix in Figure 9 (d), we found that from the first row, rotation of y-axis and translation of x-axis dominants the degeneracy. This comes from the way to build Jacobian matrix and capture the image. When the pattern of pixels with large gradients is in the middle of the image, adding the perturb to the translation on x-axis has the same affect as adding the perturb to the rotation on y-axis. So when they combine together in opposite way, the residual does not change and this leads to the "fake" degeneracy. Also for the second row, the combination of rotation of x-axis and translation of y-axis has the same effect. But this will not affect the overall degeneracy detection as shown in Figure 9 (e), because none of geometric part's degenerate directions is the linear combination of them, so it will be compensated when integrating geometric detection method.

C. Combination

From the previous section we can get different degenerate directions for both parts. However, it does not mean all these directions from two parts are degenerate since they can compensate each other in some directions. As Figure 11 shows, although there is only one plane with one line in the image which leads to three degenerate directions detected from IPP algorithm as Figure 11 (c) and three degenerate directions detected from photometric approach as Figure 11 (d), only one direction is degenerate for the system. In the Figure 11 (e), we can see the first row is dominated by the translation of x-axis which conforms to the fact.

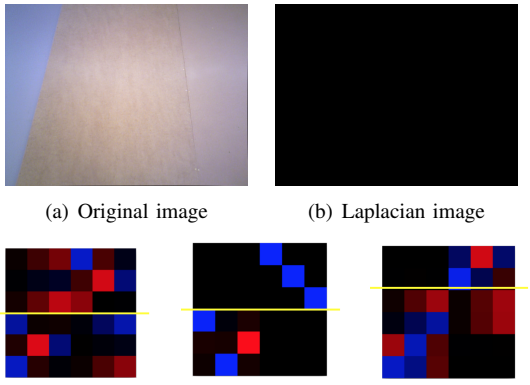
In another case, when we consider an extreme situation where there is only one whiteboard in the scene as Figure 12, both methods detect three degenerate directions and after the combination, there are still two degenerate directions which are dominated by translation of x-axis and y-axis. Without enough information provided, it is hard to localize accurately.



(a) Original image (b) Laplacian image

(c) Eigen vectors and threshold for geometric part (d) Eigen vectors and threshold for photometric part (e) Eigen vectors and threshold for overall system

Fig. 11. Totally, there is one degenerate direction which is the translation of x-axis.



(a) Original image (b) Laplacian image

(c) Eigen vectors and threshold for geometric part (d) Eigen vectors and threshold for photometric part (e) Eigen vectors and threshold for overall system

Fig. 12. When the camera only captures one blank place, it will degenerate in the translation of x-axis and y-axis for the system.

VI. DISCUSSIONS AND CONCLUSIONS

Robust to all kinds of environments, especially poorly conditioned situations, is essential to motion estimation. Once the wrong estimation is accumulated on the whole tracking, it will lead to large drift until optimization such as loop closure is applied. As the first step to robustly handle degeneracy situation, we propose a method to detect degeneracy for RGB-D odometry which bases on eigenvalue decomposition. Diverse detection algorithms are implemented for the geometric part and photometric part and then we integrate two parts to get

the overall degeneracy. The experiments on various datasets have shown the ability to detect the degeneracy.

Although the degeneracy can be detected by this algorithm, without a good initialization, updating state vector only in well-conditioned directions and ignoring the degenerate directions is not enough to solve this problem since we still do not have sufficient information. For the future work, it is possible to combine this algorithm with the factor graph optimization, which can serve as partial factors and use loop closure to refine the motion. Or add more sensors like IMU. When degeneracy happening, we can rely more on IMU measurement to avoid the degeneracy influence.

ACKNOWLEDGMENT

This work is supported by Robot Perception Lab at Carnegie Mellon University and ShanghaiTech University. The authors would like to thank all members in Robot Perception Lab for their guidance and discussions. Special thanks to Ms. Rachel Burcin, Dr. John M. Dolan and RISS team for their support.

REFERENCES

[1] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based dense planar slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5110–5117.

[2] S. Song, M. Chandraker, and C. C. Guest, "Parallel, real-time monocular visual odometry," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 4698–4705.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[4] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[5] I. Dryanovski, R. G. Valenti, and Jizhong Xiao, "Fast visual odometry and mapping from rgb-d data," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 2305–2310.

[6] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2100–2106.

[7] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visualinertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. [Online]. Available: https://doi.org/10.1177/0278364914554813

[8] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 809–816.

[9] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[10] Y. Zhou, L. Kneip, and H. Li, "Semi-dense visual odometry for rgb-d cameras using approximate nearest neighbour fields," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6261–6268.

[11] F. Steinbrcker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov 2011, pp. 719–722.

[12] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 3748–3754.

[13] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust rgb-d slam algorithm," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1714–1719.

[14] H. Cho, S. Yeon, H. Choi, and N. Doh, "Detection and compensation of degeneracy cases for imu-kinect integrated continuous slam with plane features," *Sensors*, vol. 18, p. 935, March 2018.

Deep Spatio-Temporal Video Based Analysis for Shoulder Pain Intensity Measurement

Maimoon Siddiqui[1], Diyala Erekat[1] Hamdi Dibeklioglu[2] and Zakia Hammal[3]

***Abstract*—The standard clinical assessment of pain is limited primarily to self reported pain (e.g., using Visual Analog Scale (VAS)). While self reported measurement of pain is useful, it has notable limitations including inconsistent metric, reactivity to suggestion, and susceptibility to impression management and deception. Moreover, in some circumstances self reported pain is not possible to obtain (e.g., young children and unconscious patients). Automatic facial expression analysis has emerged as a potential solution for objective, reliable, and valid pain measurement. We proposed an automatic approach for pain intensity measurement from the dynamics of facial movement. To do so, we trained automatic classifiers to measure pain intensity scores consistent with both self-reported VAS and objective Observer Pain Intensity (OPI) separately and in combination. Participants were 25 subjects with previous shoulder injury (The UNBC-McMaster Pain Archive). Participants were video recorded while they completed a series of movements of their affected and unaffected shoulders. Using the recorded videos, we investigated the contribution of spatio-temporal dynamics of facial expression for pain intensity measurement using a CNN-RNN model. From the video recordings, normalized appearance of the face was extracted using active appearance model AAM. The normalized face appearance was then fed to train an end-to-end CNN-RNN model to measure pain intensity from the video. Using two-level five fold cross-validation, the proposed model was more accurate for OPI than OPI and VAS; and each was more accurate than that for VAS only.**

***Index Terms*—Pain, Facial Expression, Visual Analogue Scale, Convolutional Neural Network, Recurrent Neural Network.**

I. Introduction

Pain is a source of human suffering, a symptom and consequence of numerous disorders, and a contributing factor in medical and surgical treatment [2]. The standard clinical assessment for pain includes using various uni-dimensional tools such as Visual Analogue Scale to record self-reported pain. While useful, self-reported pain has notable limitations including inconsistent metric properties across scale dimensions, reactivity to suggestion, and susceptibility to impression management [2]. The clinician's and patient's perception of an abstract concept of pain might not always align and could influence one another [5]. Given these limitations pain is often poorly assessed, underestimated, and inadequately treated.

[1]Maimoon Siddiqui is a senior student at School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA maimoons@andrew.cmu.edu

[1,2]Diyala Erekat, Hamdi Dibeklioglu are with Computer Engineering Department, Bilkent University, Ankara, Turkey diyala.erekat@bilkent.edu.tr, dibeklioglu@cs.bilkent.edu.tr

[3]Zakia Hammal is a Senior Project Scientist at Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA zhammal@andrew.cmu.edu

With the release of publicly available pain database and advancements in computer vision and machine learning, automatic pain assessment from facial expressions has emerged as a possible solution for automatic and objective assessment of pain [2]. Most previous efforts for automatic pain assessment have focused on frame-level pain intensity measurement consistent with Prkachin and Solomon Pain Intensity (PSPI) scores, which codes movement of facial action units from static facial images (for a detailed description of the state of the art efforts for automatic pain assessment please see [2]). Despite VAS being gold standard in clinical practice, previous efforts focused on static frame-level pain estimation. One exception is Liu and colleagues [1] that proposed a personalized model for automatic assessment of VAS pain scores from videos. The authors combined facial shape with predefined personalized features (i.e., age range, gender, complexion) to train an end-to-end combination of neural network and Gaussian Regression model for VAS pain intensity.

However, Liu and colleagues [1] explored pain related spatial information in static images and did not investigate the contribution of dynamic changes of facial expressions in consecutive frames. Pain communication is dynamic. Dynamics of pain warrants explicit attention because it furthers our understanding of the pain experience [2]. As a contribution to previous efforts, we propose a spatiotemporal end-to-end CNN-RNN model for pain intensity measurement in video sequences. Because some gradations of pain intensities were sparsely represented, a stratification technique was performed during the training to improve the model performances. Stratification divided the data into folds for training, validation and testing such that each division ensured the classifier saw each pain intensity level at least once for each fold.

II. Methods

The goal of this paper is to automatically estimate pain intensity from the spatiotemporal changes in facial expression during pain video sequences.

A. Face and Facial Features Extraction

The first step in automatic pain intensity measurement from facial expression is the automatic detection of face and facial features in video sequences. To do so, we used the 66 facial points tracked using Active Appearance Model (AAM) and distributed with the database (see Fig.1) [12]. The tracked facial points were then used to extract the appearance of the face from each video frame (see Fig.3). To remove non-rigid head variation, tracked faces were registered to a reference

face using Delaunay Triangulation[1] as described in sections B and C.

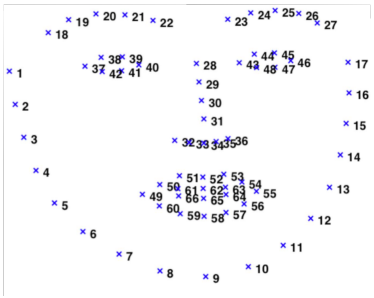


Fig. 1: 66 facial landmark points

B. Average Facial Landmarks Computation

Initially, the input frames in the video sequences were scaled and centered to a predefined inter pupillary distance. This was done because the input frames were not of the same dimension and the faces in those images were not frontal and centered at the same height on the frame. The scaled ad centred facial frames were then used to find the average facial landmark points.

As a first step to normalize and center images, all the input images were transformed to output images of fixed size (e.g., $(w \times h)$ as (1500×1500)).

The (x,y) coordinates were available for all 66 facial points $L = [(x^1, y^1), (x^2, y^2)...(x^{66}, y^{66})]$. The input images were transformed so that they were centered to a fixed size (e.g., $\frac{1}{3}$ of the height of the output images), maintaining a fixed inter-pupillary distance (e.g., 600 pixels) in all the output images of the predefined face size (i.e., $(width \times height)$ as (1500×1500)). This was done by aligning the landmark points # 36 and # 45 associated with the corner of the left and right eyes to be transformed from:

$$src_1 = \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} \quad \rightarrow dst_1 = \begin{bmatrix} \frac{3}{10} \times w \\ \frac{1}{3} \times h \end{bmatrix} \tag{1}$$

$$src_2 = \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} \quad \rightarrow dst_2 = \begin{bmatrix} \frac{7}{10} \times w \\ \frac{1}{3} \times h \end{bmatrix} \tag{2}$$

where (x^1, y^1) are the input coordinates of the landmark points of corner of the eyes.

A transformation matrix is then found to map the source coordinates to the destination coordinates.

$$f : transform\_matrix = (src, dst) \tag{3}$$

All the remaining landmark points L were then transformed with the transformation matrix and an affine transformation is performed on the input image with the transform matrix $f(Image)$

[1]https://www.learnopencv.com/average-face-opencv-c-python-tutorial/

(a) Original Image **(b)** Centered Image

Fig. 2: The figure shows how the frames were initially registered.

The resulting images then have the same dimension and are centered on the same height. The landmark points for the average face were found by taking an average of all the landmark points in the centered, normalised input frames.

$$\sum_{l=1}^{L} \frac{1}{F} \sum_{f=1}^{F} (x_L, y_L) \tag{4}$$

where F is the total number of frames and (x,y) are the coordinates of each of the 66 landmark points in L.

C. Warping centered images to the Average Landmarks using Delaunay Triangulation

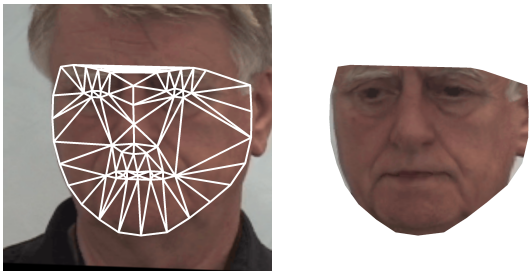
After the computation of average landmarks, the pixels of each video frame were warped using Delaunay Triangulation. The 66 landmark points were used to perform the Delaunay triangulation that divides the average face landmark points into the triangles as shown in Fig 3. This is done so that later similarity transform could be performed for every pair of triangle to warp input images to the average facial landmark points.

The 66 landmark points were divided into triangles where each triangle is a list of three points. For every triangle in average facial landmark points and every corresponding triangle in the input image, an affine transformation matrix was needed to perform the transformation of the three corners of the triangle of the input image to the three corners of the triangle of the average facial landmark points. Delaunay Triangulation was used to find the transformation matrix from all corresponding triangles in the input image to the average facial landmark points. The learned transformation matrix is then used to wrap the entire image triangle by triangle.

The faces are then cropped out by forming a mask with the convex hull of the landmark points (see Fig.3.b).

D. End-to-End Spatio-Temporal Pain Intensity Measurement

The warped face images were used to train a spatio-temporal model to estimate pain intensity from video sequences (see Fig. 4). Convolutional Neural Network (CNN) was trained to learn the spatial features from the video frames followed by a Recurrent Neural Network (RNN) trained to learn the dynamics changes in the extracted spatial features between consecutive video frames. Unlike most state of the art work



(a) Triangulation applied on facial landmarks. **(b)** Face warped to the average face.

Fig. 3: Face registration using Delaunay triangulation; (a), (b).

that models automatic pain assessment as a classification task, we used regression to estimate pain intensity scores.

1) Spatial Representation: CNN is a feed forward deep neural network that learns spatial features from the images with regard to the output task through back propagation of the loss function [11]. CNN allows networks to be deep with reduced number of learnable parameters by sharing weights in a layer. We used the pre-trained AlexNet[2]. AlexNet was used as a feature extractor at the frame-by-frame level (the last fully connected layer, 4096D feature). AlexNet was fine-tuned when the end-to-end spatio-temporal model was trained (and the weights were updated by continuing the back propagation [9]) to estimate pain intensity scores from the video.

2) Temporal Representation: RNN are neural network where outputs from previous steps are fed as inputs to current step. This allows RNN to have "memory" of past information. Therefore, RNN was used because of their successful learning of sequential data. RNN was trained to learn the temporal changes in the spatial information learned through CNN. To do so, the 4096D feature vectors from consecutive video frames outputted from the CNN were directly fed into the RNN. To remember possible long-term temporal dependencies we used Gated Recurrent Network [13] (GRU). GRU works similarly to LSTM without an additional forget gate and so have fewer parameters to learn and a simpler architecture which made it more suitable and computationally easier for small database (200 videos in our case). That ensured that we do not over fit and still generalize well. This also helped us to tackle the problem of the gradient vanishing over long sequences of videos. We used a 2 layers GRU model with 64 hidden units.

The output $X = \{x_1, x_2..., x_{64}\}$ of the RNN was passed through a multivariate linear regression layer to estimate pain intensity scores Y as shown in equation 5

$$\mathrm{Y} = W \times X + B \tag{5}$$

where X is the $64 \times n$ output vector from the RNN being fed into the multivariate linear regression layer, W is 1×64 the weight coefficient vector for regression, B is the bias term and Y is the pain intensity score.

[2]https://download.pytorch.org/models/alexnet-owt-4df8aa71.pth

The number of layers in RNN, and other hyper parameters for the proposed model were determined by minimizing the validation error during the training process. The different hyper parameters tuned during the process are shown in Table 1.

TABLE I: The different hyper parameters that were tuned during the training process

Hyperparameter	Values
Number of hidden layers	{2,3,4}
Learning rate	{0.00001,0.0001}
Dropout factor	{0.1,0.2,0.3}
Regularization rate	{0.000001,0.000005}
α custom loss factor	{0.3}

The best parameters were chosen from here

III. Database

Automatic pain assessment has many limitations and availability of well annotated pain databases is one of them. Training a deep model requires a large volume of reliably annotated data. Data annotation is a tedious task and there is a lack of proper training and availability of public databases specifically for pain. The UNBC McMaster Pain Archive [12] was the first effort to address the needs for well annotated facial expression database for acute pain [2].

The Pain Archive database consists of 129 participants (63 males and 66 females) who self-identified as having shoulder pain and were recruited from physiotherapy clinics and through advertisements [12]. Participants were video recorded during different exercise routines to test their affected and unaffected shoulder. The archive which is publicly available consists of 200 video sequences from 25 different participants (with 48398 frames [12]). For each video sequence, the data contains three self-reported pain scores and one observer pain intensity score. After each test, participants self-reported the maximum pain they experienced using three different scales [12]. The first two self reported scales were sensory scale (SEN) and affective motivational scale (AFF). Those two were performed on a 15 point likert-type scale which ranged from 0-14. SEN ranged from "extremely weak" to "extremely insane" and AFF ranged from "bearable" to "excruciating". The third self reported scale was Visual Analogue Scale VAS which was a 10 cm long scale anchored at each end with words "No pain" and "Pain as bad as it could be". Additionally, a fourth independent and offline observer pain intensity (OPI) was collected from the recorded videos. OPI was measured on a 6 point likert-type scale that ranged from "no pain" to "strong pain". Finally, each of the 48398 video frames had 66 facial landmark points tracked using AAM (see Fig.1). Fig. 5 shows the data distribution of Pain Archive per pain intensity per label.

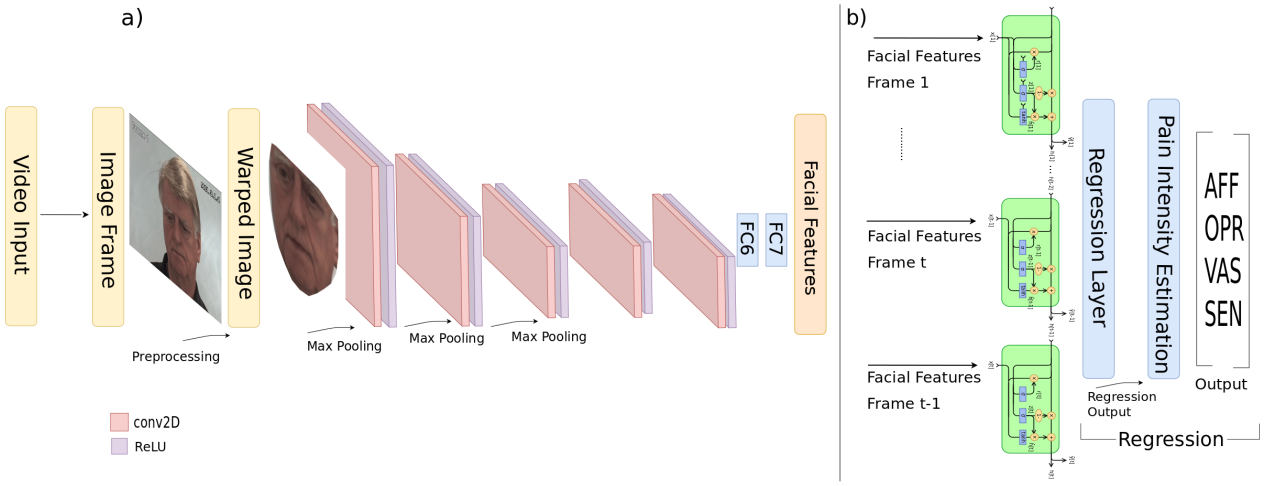


Fig. 4: The architecture of the proposed model a) The Convolutional Neural Network: AlexNet trained to learn frame-by-frame spatial feature (4096D per frame and b) Recurrent Neural Network: 2-layer GRU4 trained to learn per-video temporal dynamics of facial features.

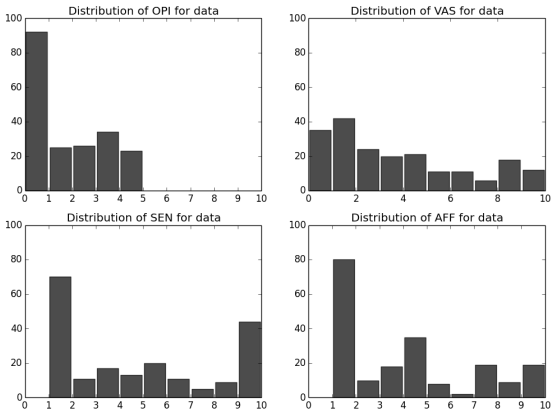


Fig. 5: Pain intensity distribution across the 4 scales for the entire database. VAS scale 0-10, SEN scale 0-10, AFF scale 0-10,(where any value above 10 has been clipped off as an outlier), OPI scale 0-5.

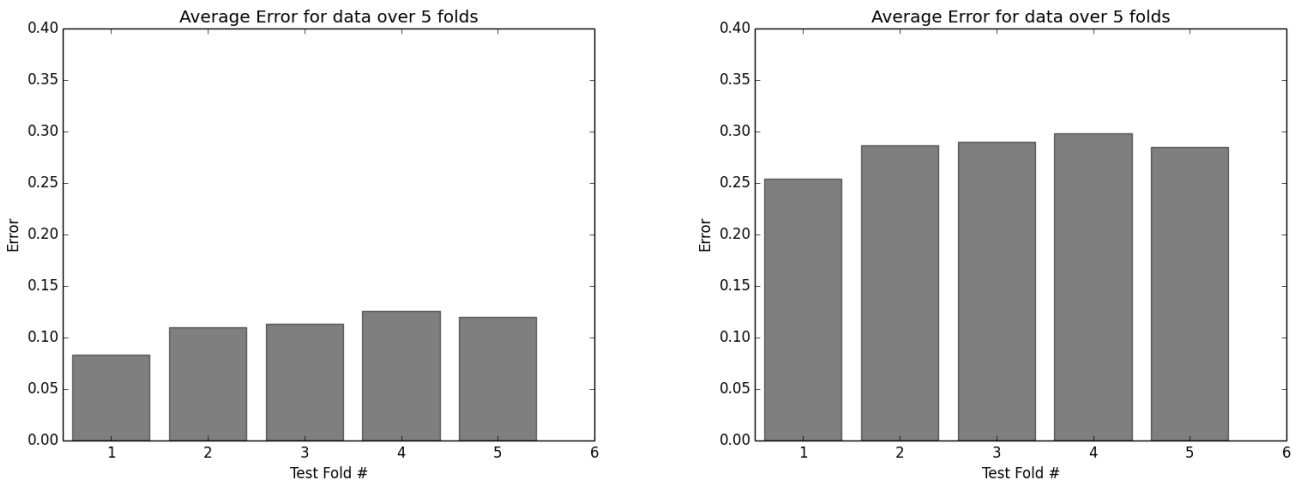
IV. Experimental Results

Two level nested cross-validation was used to test the proposed spatiotemporal model for pain intensity measurement (see Algorithm1). That included running an outer cross validation for splitting the data into test and training folds. Then running the inner loop to select the best parameters based on minimum validation loss (see Table 1) through another cross validation on the training fold.

In the outer 5 folds cross validation, the data was divided into 5 independent subsets. For each run of the outer cross validation, one subset was kept as test-set and the rest were provided to the inner loop for training and validation. In the outer loop, every subset got to be part of testing exactly once using the remaining 4 subsets for training and validating. Every test value was an independent evaluation of the model and an average of those test errors represented how the model would generalize. After the outer test holdout subset, the remaining folds were passed onto the inner loop. In the inner loop every subset got to be part of validation exactly once, using the other remaining folds as training. The inner cross validation was used to train and search for best hyper parameters. The hyper parameters tuned were: the drop out factor, the learning rate, and the λ regualrizer rate for the loss function.

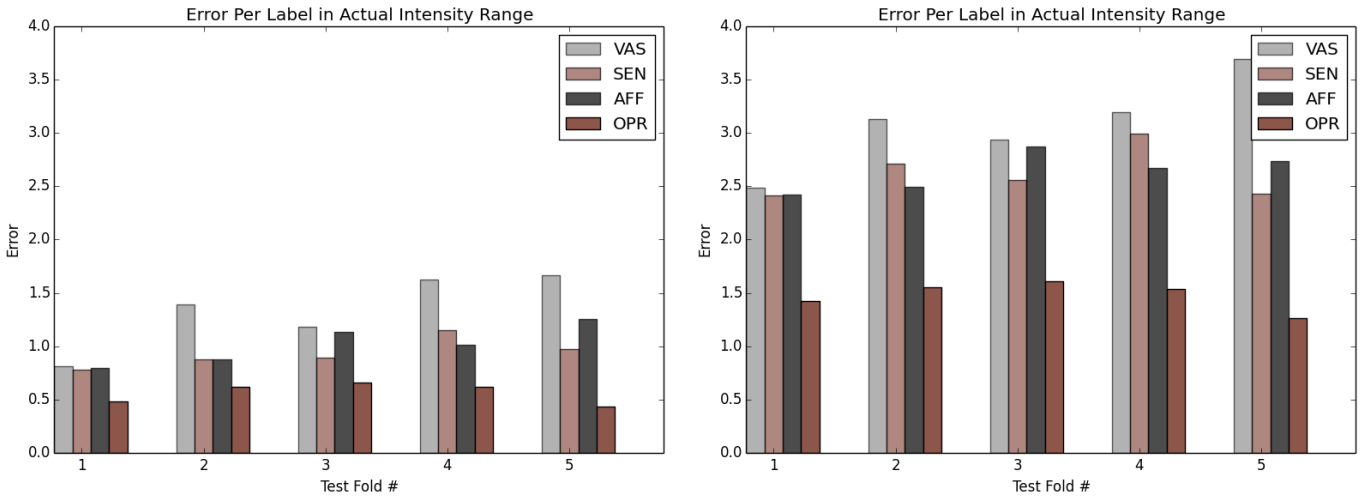
Data: Pain Archive
Result: Test loss for 5 folds
params = [num layers, dropouts, learning_rate, λ]
for *outer_fold (5)* **do**
    min_valid_loss = infinity;
    test_data = data[outer_fold];
    train_val_data = data - test_data;
    for *inner_folds* **do**
        val_data = data[inner_fold];
        train_data = train_val_data - val_data;
        net.init(params[inner_fold]);
        net.fit(train_data);
        valid_loss = net.history['valid_loss'];
        if *valod_loss* $<$ *min_valid_loss* **then**
            net.save_params(params[inner_fold]);
            min_valid_loss = valid_loss;
            best_param_idx = params[inner_fold];
        end
        test_net = net.init(params[best_param_idx]);
        test_net.fit(train_val_database);
        test_loss = evaluate(test_net,test_data);
    end
end

Algorithm 1: Nested two level cross validation



(a) MSE error. The range of the normalized training score is [0-1]

(b) MAE error. The range of the normalized training score is [0-1]



(a) MSE error per intensity label in original range

(b) MAE error per intensity label in the original range.

Fig. 7: (a) Mean squared error (MSE) (b) Mean Absolute Error (MAE) for the 5 test folds of the stratified distribution of the database. The original range: VAS scale 0-10, SEN scale 0-10, AFF scale 0-10, OPI scale 0-5

A. Loss Function and Test Metric

The proposed model predicts pain intensity labels. We evaluated the significance of the self reported pain labels and observer rating by training different models. The models were trained on individual pain labels and different combination of pain labels (VAS alone; OPI alone; OPI and VAS; AFF, OPI, VAS, SEN).

A custom loss function (see equation 6) with an inter label variance factor and regularization (β) was used to train the CNN-RNN model. The custom loss was used for models trained on combined pain labels. The three self reported labels including VAS, AFF and SEN are highly co-related and the only observer reported label which is OPI has lower correlation to the other three labels. Looking at the correlation between the labels, we wanted our model to learn from both the self-reported and observer reported labels. To counter the problem of low correlation between the two types of reported pain and penalize the difference between the two, an additional term for the inter class variability $Var(\hat{Y})$ was added. This ensured that the model tried to minimize the inter class variability and this Gaussian noise between some uncorrelated labels was reduced. This modification of the loss function ensured that we only identified meaningful minute differences and learned a better relationship between the data and the labels.

$$CustomLoss = \frac{\alpha}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + (1-\alpha)Var(\hat{Y}) + \lambda\sum_{p=1}^{P}(\beta_p) \tag{6}$$

Where n are the number of instances the loss was averaged

over, y_i is the observed pain intensity vector and $\hat{y}_i$ is the predicted pain intensity vector and $Var(\hat{Y})$ is the inter class variability term added to custom loss with a multiplicative factor of α. For individual pain scores estimation (e.g., VAS alone), α was set to 1, in the case of combined pain scores optimization α was chosen during the CNN-RNN training process. To help the model generalize better on unseen test data, the custom loss equation has an additive factor for L2 regularization where β is the regularization term.

We evaluated the model with Mean Absolute Error and Mean Square Error while testing for generalizability (see equations 7 and 8 and Table II). During the test phase, the obtained error scores were rescaled from 0-1 to their original range of 0-10 and 0-5 for AFF, VAS, SEN and OPI, respectively. Normalized mean errors were used to assess similarity between individual folds with the entire database for each of the 4 pain intensity measures individually and in combination.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

TABLE II: The average test error comparison between random and startified distribution of data of VAS and OPI in 4 settings.

Pain Scale	Random Distribution		Stratified Distribution	
	MAE (VAS)	**MAE (OPI)**	**MAE (VAS)**	**MAE (OPI)**
VAS	2.98	-	2.57	-
OPI	-	1.46	-	1.33
VAS, OPI	2.06	1.63	2.25	1.72
VAS, OPI,AFF, SEN	3.09	1.48	2.54	1.39

B. Stratified distribution of data

Because some gradations of pain intensity were sparsely represented, we performed a stratification to distribute the database into 5 folds of well balanced pain intensity scores per fold in the two-fold cross validation process. This was to ensure that the distribution of the classes in the training folds was well balanced leading to a better generalization to unseen data. To do so, for each of the 4 pain intensity metrics, the frequency F = $\{f_1, f_2....f_5/f_{10}\}$ of the intensities (0-10 for VAS, SEN and AFF, and 0-5 for OPI) were normalized. Then mean square error was computed between these normalized frequencies with the normalized frequencies of the entire database D = $\{d_1, d_2....d_5$ or $d_{10}\}$. The similarity was calculated as 1 minus the normalised mean square error in equation 9.

$$1 - \sum_{i=1}^{I}(d_i - f_i)^2 \tag{9}$$

TABLE III: The similarity of data distribution of the 5 folds to the entire database

Fold #	AFF	VAS	OPI	SEN
Fold 1	97.47	97.6	93.86	97.95
Fold 2	98.93	99.42	99.45	98.41
Fold 3	97.89	98.28	98.33	97.86
Fold 4	98.39	98.28	99.25	97.96
Fold 5	93.92	96.79	98.37	96.23

[a]Mean square similarity with the entire Pain Archive

The test error results for average and each pain intensity across the five folds found with this stratification technique are shown in Fig 7. Results are reported using MSE (Fig. 7(a)) and MAE (Fig. 7(b)). The test errors across 5 folds for both MSE and MAE have high consistency due to the balanced stratification of the folds. We trained our model on stratified distribution with custom loss and regularization. We used four different settings in which we trained on 1) One pain intensity level-VAS 2) One pain intensity level-OPI 3) Two pain intensity levels- VAS,OPI and 4) All four pain intensity levels- AFF, OPI, VAS, SEN. Table III shows a comparison between the average MAE and MSE test errors for random and stratified distribution strategies. Better results were found using stratified distribution.

C. Comparison With the State of the Art

We compared our model to DeepFaceLift (LIFT = Learning Important Features) [1], which is a two staged personalized deep neural network. DeepFaceLift is the only available model for self reported pain intensity measurement from videos. The first stage DeepFaceLift model is a fully connected neural network that takes AAM facial landmarks as input. The second stage is a Gaussian Processs Regression model that takes the output of the first stage as input and outputs VAS scores. The two settings which we compared to were when personal features were inserted in 1) 3rd Neural Net Layer and 2) Neural Net Input. For comparison purpose with DeepFaceLift, the model was tested with MAE which measures the average absolute distance between the estimated and true values.

Method	Setting	Pain Scores	Output VAS MAE
Our Model	**One label**	**VAS**	**2.57**
	One label	**OPI (Output:OPI)**	**1.33**
	Two labels	**VAS,OPI**	**2.25**
	Four labels	**AFF, OPI, VAS, SEN**	**2.54**
DeepFaceLift	3rd Neural Net Layer	VAS,OPI	2.34
		VAS,OPI/VAS	2.18
		VAS	2.24
	Neural Net Input	VAS,OPI	2.41
		VAS,OPI/VAS	2.22
		VAS	2.22

TABLE IV: Error comparison with DeepFacelLift [1] with Mean Absolute Error in the original scale of 0-10 (0-5 in the case of OPI)

The lowest reported MAE for VAS which is 2.25 (See Table IV) is comparable to the state of the art DeepFaceLift. While they passed on a summary measure of the video

sequences to their regression layer as well as personalized features, we only explored the temporal dynamics of the videos. This comparable result shows how temporal dynamics convey important and equally beneficial information for the course of pain experience. I addition, we reported for the first time in the state of the art results on OPI pain intensity measurement.The lowest error reported was 1.33 (for a range of 0-5) for OPI alone, which indicates that OPI offers more reliable, and objective pain assessment.

V. Conclusion

We proposed a spatio-temporal approach for pain intensity measurement of self reported and observer's reported pain intensity scores from video sequences. CNN was used for spatial representation and RNN for temporal information in the course of a pain experience. For future work, we will test the generalizability of our model on a larger database, and will investigate adding facial shape movement in addition to the appearance.

Acknowledgment

This project was part of a 2 months Robotics Institute Summer Scholar Program at Carnegie Mellon University and in collaboration with Bilkent University. Research reported in this paper is an internal report of trainees progress and is still work under progress for further evaluation and validation. Effort was supported in part by the National Institute of Nursing Research of the National Institutes of Health under Award Number R21NR016510. The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsors

References

[1] D. L. Martinez, O. Rudovic, and R. Picard, "Personalized Automatic Estimation of Self-Reported Pain Intensity from Facial Expressions," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017.*

[2] Z. Hammal and J. F. Cohn, "Automatic, Objective, and Efficient Measurement of Pain Using Automated Face Analysis," *Social and Interpersonal Dynamics in Pain, pp. 121–146, 2018.*

[3] Z. Hammal and J. F. Cohn, "Towards Multimodal Pain Assessment for Research and Clinical Use," *Proceedings of the 2014 Workshop on Roadmapping the Future of Multimodal Interaction Research including Business Opportunities and Challenges - RFMIR 14, 2014.*

[4] M. Bellantonio, M. A. Haque, P. Rodriguez, K. Nasrollahi, T. Telve, S. Escalera, J. Gonzalez, T. B. Moeslund, P. Rasti, and G. Anbarjafari, "Spatio-temporal Pain Recognition in CNN-Based Super-Resolved Facial Images," *Video Analytics. Face and Facial Expression Recognition and Audience Measurement Lecture Notes in Computer Science, pp. 151–162, 2017*

[5] M. A. Haque, R. B. Bautista, F. Noroozi, K. Kulkarni, C. B. Laursen, R. Irani, M. Bellantonio, S. Escalera, G. Anbarjafari, K. Nasrollahi, O. K. Andersen, E. G. Spaich, and T. B. Moeslund, "Deep Multimodal Pain Recognition: A Database and Comparison of Spatio-Temporal Visual Modalities," *2018 13th IEEE International Conference on Automatic Face And Gesture Recognition (FG 2018), 2018.*

[6] C. Jonassaint, N. Rao, A. Sciuto, G. Switzer, L. D. Castro, G. J. Kato, J. H. Jonassaint, Z. Hammal, N. Shah, and A. Wasan, "A Cross-sectional Feasibility Study Testing the Use of Abstract Animations for the Communication and Assessment of Pain (Preprint)," 2018.

[7] P. Rodriguez, G. Cucurull, J. Gonalez, J. M. Gonfaus, K. Nasrollahi, T. B. Moeslund, and F. X. Roca, "Deep Pain: Exploiting Long Short-Term Memory Networks for Facial Expression Classification," IEEE Transactions on Cybernetics, pp. 1–11, 2017.

[8] Z. Chen, R. Ansari, D. Wilkie, "Automated Pain Detection from Facial Expressions using FACS: A Review", 2018. https://arxiv.org/abs/1811.07988

[9] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks", Proceedings of NIPS, IEEE, Neural Information Processing System Foundation, pp. 1097-1105, 2012.

[10] R. Yang, S. Tong, M. B. Lopez, E. Boutellaa, J. Peng, X. Feng, A. Hadid, "On Pain Assessment from Facial Videos Using Spatio-Temporal Local Descriptors", 2016.

[11] W. Pei, H. Dibeklioglu, T. Baltrusaitis, D. M. J. Tax, "Attended End-to-end Architecture for Age Estimation from Facial Expression Videos", 2017.

[12] P. Lucey, J. Cohn, K. Prkachin, P. Solomon, I. Matthews. "Painful data: The UNBC-McMaster shoulder pain expression archive database", IEEE International Conference on Automatic Face and Gesture Recognition and Workshops, FG 2011. pp. 57-64, 2011.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", NIPS 2014 Workshop on Deep Learning, December 2014.

Adaptive Multimodal Fusion for Grasping Transparent and Specular Objects*

Yimin Tang[1,3] Thomas Weng[2] David Held[2]

***Abstract*— Intelligent grasping robots can help humans in many situations like our homes, hospitals, and factories. State of the art methods for grasping use depth images captured by depth sensors like time-of-flight, structured light, and stereo cameras. Depth images are effective for grasping opaque objects; however, they are problematic for transparent and specular objects, like the ones in Fig. 1. These objects do not appear in depth images, but do appear in RGB images. To improve grasping performance on transparent and specular objects, we learn a network that takes RGB images as input from an existing depth image network, and then combine the two networks together. We achieve a 60.0%, 50.7% success rate for transparent and specular objects compared to the state of the art(FC-GQCNN) 20.0%, 49.3%.**

I. INTRODUCTION

One of the fundamental problems for robotics is object grasping. Many scenarios can make use of robotic grasping to help people complete tasks. For example, we can use robots to grasp medicine for motor-impaired individuals or sort products in supermarkets. However, the diversity in object materials and geometries makes robotic grasping a difficult task.

Many existing grasping methods rely on depth images to plan grasp strategies. There are many ways to generate depth images like 1) time-of-flight (ToF), 2) structured light, and 3) stereo cameras. ToF cameras send pulses to calculate each point distance of an image like radars [24]. Structured light cameras project light patterns to get depth images [23]. Stereo cameras are a particular case of light fields and use two standard parallel cameras' figures to calculate final depth images by comparing disparities [11], [22]. The first two methods have excellent performance for opaque objects in indoor environments. But for transparent and specular objects, these three depth detection methods fail to get adequate depth information. These failures can occur as incorrect depth values in the case of transparent objects, and as missing depth readings for specular objects [9]. Existing depth-based grasping methods fail to grasp objects with these properties.

In this paper, we propose a new approach that adds RGB images and an RGB-based model to state-of-the-art depth-based model to achieve robust grasping performance on transparent and specular objects. We provide a new method for robust grasping on transparent and specular objects based on existing depth-based and RGB-based models. Our goal is to combine the output of the RGB and depth models by training a third model that weights their grasp prediction outputs. For example, for opaque objects in normal light conditions, we want the model more rely on depth models in the image regions because, in these conditions, depth-based models have better performance. And for transparent objects, the depth images are not good enough that objects are invisible. And ideally, the combined model will give the RGB models more priorities. We evaluate our method on transparent, specular, and opaque objects and show that it outperforms existing methods.

*This work was supported by Carnegie Mellon's Robotics Institute Summer Scholars (RISS) program, Robots Perceiving and Doing(R-Pad) Lab, Carnegie Mellon University and ShanghaiTech University

[1]ShanghaiTech University tangym@shanghaitech.edu.cn
[2]Carnegie Mellon University tweng,dheld@andrew.cmu.edu
[3]Carnegie Mellon's Robotics Institute Summer Scholars (RISS) program

II. RELATED WORK

A. Grasping

Robotic grasping is a challenging area for robotics and has been studied extensively by many researchers.[2], [1], [14], [19]. Grasping methodologies are divided into two main categories: analytic and empirical[16]. Analytic approaches often use physics-based contact models choose a grasp, but these methods are both computationally expensive and sensitive to dynamics like contact friction. Recent advances in grasping make use of data-driven methods to achieve better grasping performance [1], [14], [5]. Data-driven models are used to capture uncertainties of object parameters like shape and friction[17]. However, these depth-based data-driven models fail to detect transparent and specular objects which we consider. Our recent work demonstrates that learning an RGB-based model from an existing depth-based model can achieve better performance on transparent and specular objects [21].

This work attempts to combine the output of these two models to boost grasping performance beyond either of them individually. For this work, we use a very similar representation of the Fully Convolutional Grasp Quality CNNs (FC-GQCNN) [17] which is a state of the art depth-based model. Our network architecture is shown in Fig. 3

FC-GQCNN learns a function $G(q_d, I_d)$ and parameterizes q_d as (x, y, θ, z), where x and y represent the grasp point in the planar coordinates, θ is the angle of the gripper and z is the grasp depth relative to the depth camera. FC-GQCNN can output dense predictions $G(q_d, I_d)$ across the entire depth image because of the fully convolutional design. (We will describe the notations in APPROACH section)

B. Adaptive modality fusion

Sensor fusion has been explored for many robotics domains like self-driving cars and object detection. Hand-

(a) Grasping scene (b) RGB image (c) Depth image

Fig. 1: Transparent and specular objects provide poor depth readings with conventional depth sensors, posing a challenge for depth-based grasping techniques. (c) Most values in the depth image are close to the table. Sawyer robot employing proposed method to grasp transparent objects.

engineered mixtures of experts were proposed by Enzweiler *et al.* [4] prior to the advent of data-driven models. Premebida *et al.* [15] trained a fusion SVM which used designed features to combine RGB and depth models. Late-fusion networks have been used in computer vision tasks like RGB-D object recognition[3], [6] and multimodal pedestrian detection[18], [20]. Oier Mees *et al.* [13] propose a method that does not use any prior information when the model learns fusion weights and also can be used with manually designed fusion features. In comparison to our fusion method, we keep the main architecture and change the loss function, which only updates for one single point.

III. APPROACH

In this section, we describe our approach in detail for fusing RGB and depth models. Our model architecture is shown in Fig. 2

A. Problem Statement

We define the input modality $\mathcal{M}_d$(such as depth) for a grasp. We assume that this grasping method will output a grasp score $G(q, I_d)$ when we give a grasp q and an image I_d such as a depth image. We wish to solve two tasks. The first task is that we wish the knowledge based on modality $\mathcal{M}_d$ can be transferred to $\mathcal{M}_s$ when given an Image I_s on a new modality $\mathcal{M}_s$ (such as RGB). The second is the following: assume we have another model and we wish this model will output weights $W_d(I_d, I_s)$ and $W_s(I_d, I_s)$ when we give two images I_d on $\mathcal{M}_d$ (such as a depth image) and I_s on $\mathcal{M}_s$ (such as a RGB image).

We assume each input for the second task is a pair of two images (I_d, I_s). We assume that all images in each pair were taken simultaneously and at the same resolution from the two modalities $\mathcal{M}_d$ and $\mathcal{M}_s$.

B. Supervision Transfer for RGB model

This work has been done by [21], and we will describe it in details for the consistency of the following fusion method. We can quickly notice in Fig. 1 that two types of images (RGB and depth image) have complementary strength. It means that we have a big chance to get enough information from one picture if the other can not provide enough sufficient information. For example, we can get more useful information from RGB images for a glass cup when there are many resulting noise and missing data in our depth images. And highly textured objects provide more difficulties for RGB-based model, but we can ignore texture through depth images.

Weng *et al.* [21] use supervision transfer [7], [8], [12] to train a model for modality $\mathcal{M}_s$ (such as RGB). Their dataset D' uses opaque objects that depth-based grasping methods typically perform well. For each paired image I_d, I_s, they compute the score $G(q, I_d)$ for $\mathcal{M}_d$. And then train a model whose output we can note as $G_\phi(q, I_s)$ for the modality $\mathcal{M}_s$. If we use parameter ϕ to parameterize the model, their model's loss function can be written as

$$\mathcal{L}(\phi) = ||G(q, I_d) - G_\phi(q, I_s)||^2 \tag{1}$$

C. Supervised Learning for Multimodal Fusion

To train a multimodal fusion model, we collect real grasping data using a naive policy of taking the simple average of the RGB and depth model outputs, a method we call RGBD-M. Then we choose the grasp with max success probability for the averaged output. There are other heuristics which may perform better for collecting real grasp such as using maximum between two outputs.

$$G_\phi(q, I_d, I_s) = 0.5 \cdot (G(q, I_d) + G_\phi(q, I_s)) \tag{2}$$

Using this model we can get a dataset D. For each grasp attempt, we have $G(q, I_d)$, $G_\phi(q, I_s)$, I_d, I_s, and the result label t of RGBD-M grasping. We wish to use the result label to train our multimodal fusion. We can calculate the final result with multimodal fusion:

$$G_{w\phi}(q, I_d, I_s) = W_d(q, I_d)G(q, I_d) + W_s(q, I_s)G_\phi(q, I_s) \tag{3}$$

$$W_d(\cdot, \cdot), W_s(\cdot, \cdot) \in [0, 1] \tag{4}$$

$$W_d(q, I_d) + W_s(q, I_s) = 1 \tag{5}$$

Then we define the loss function:

$$\mathcal{L}(\phi) = \begin{cases} cross\_entropy(G_{w\phi}(q, I_d, I_s), t) & q = q_t \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$q_t = argmax_q(G_{w\phi}(q, I_d, I_s)) \tag{7}$$

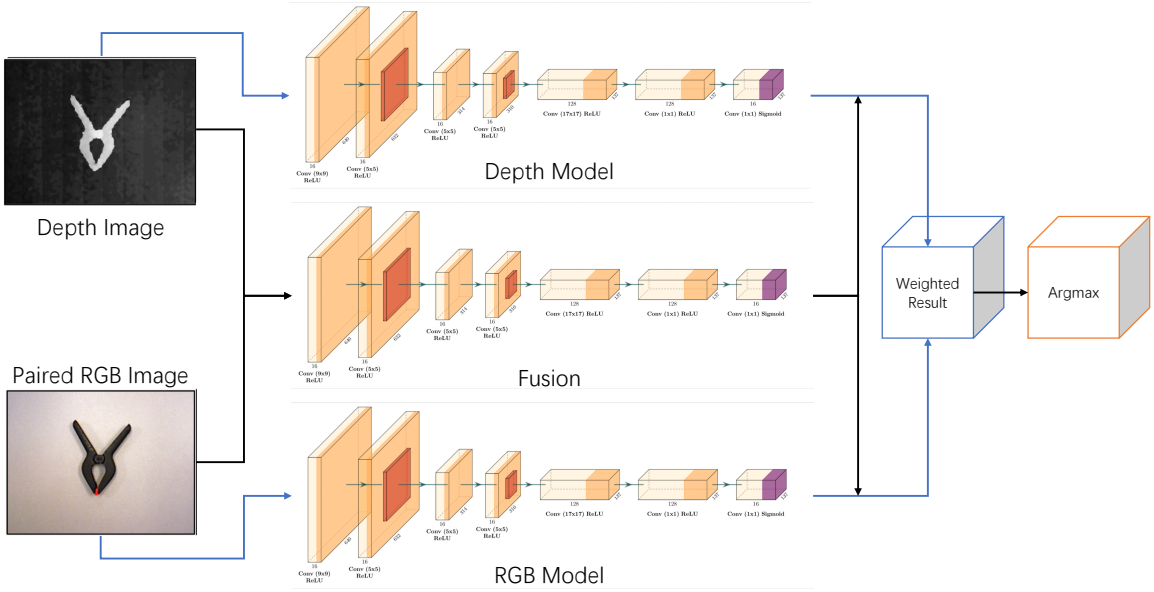


Fig. 2: The whole picture of our multimodal fusion(RGBD-W)

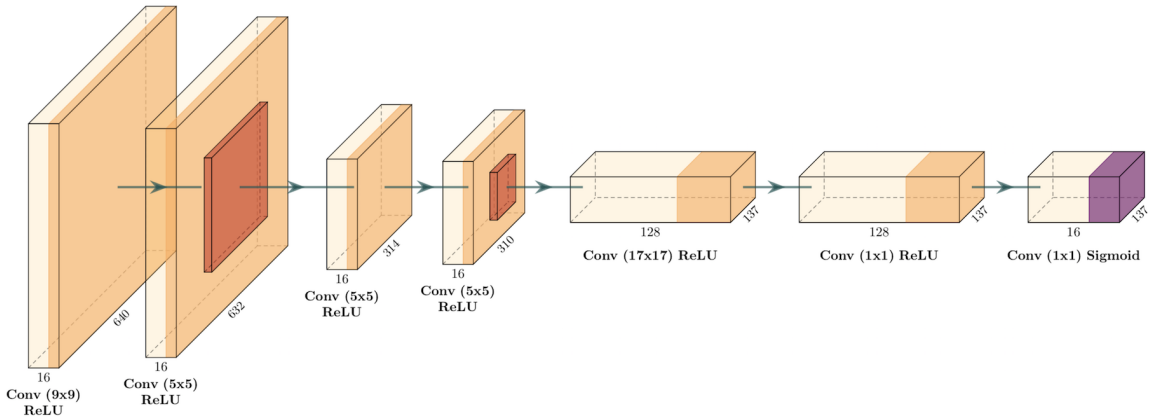


Fig. 3: Network architecture. The output is a 3D array of dense grasp predictions over (x, y, θ). Compared to FC-GQCNN, we modify the first layer to allow the input to be either 3 or 4 channels (RGB or RGB-D) rather than just a single channel as in the original depth-based network.

D. Implementation of Multimodal Fusion

For this work, we use a very similar representation of FC-GQCNN. Because the RGB modality does not provide depth information as output, we change $q = (x, y, \theta, z)$ to $q = (x, y, \theta)$ by using the maximum z in each bin of the depth output. Using this specification, a robot gripper starts above the surface with objects and lowers itself until it collides with either the surface or an object. We also modify our loss function slightly:

$$q = max_z\left((x, y, \theta, z)\right) \tag{8}$$

$$\mathcal{L}(\phi) = \begin{cases} cross\_entropy(G_{w\phi}(q, I_d, I_s), t) & q = q_t \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$q_t = argmax_q(G_{w\phi}(q, I_d, I_s)) \tag{10}$$

Then we train the fusion model in Fig. 4 so that for a failed grasp label, the fusion model will lower the weight of the model that has calculated a higher probability at failed grasp point.

IV. EXPERIMENTAL RESULTS

Our experiments are designed to answer the following question: How does the performance of our fusion model

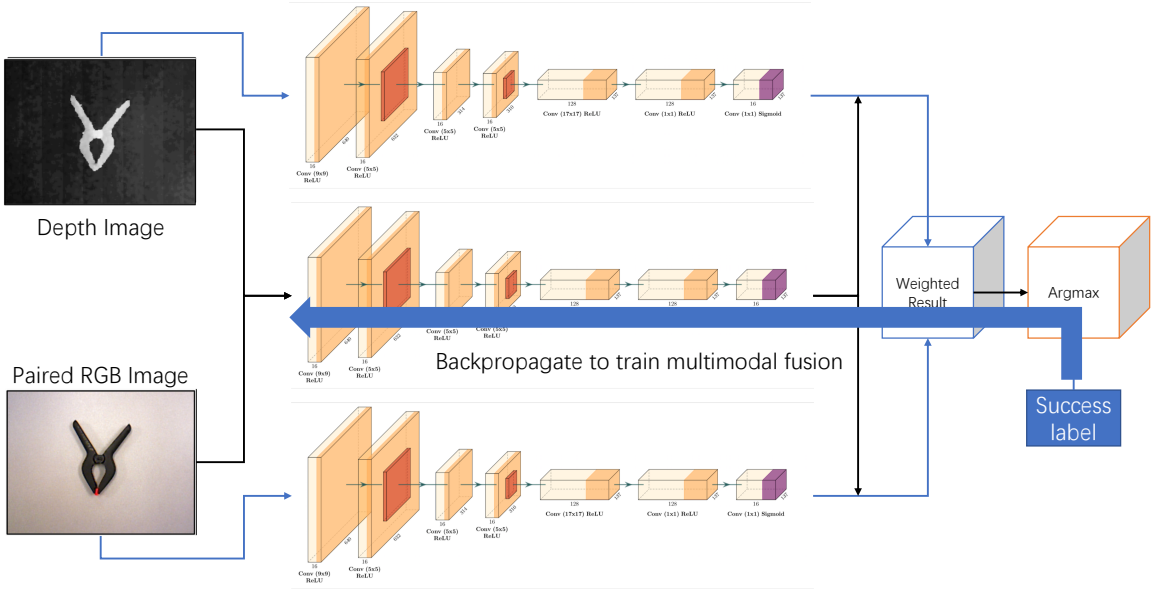


Fig. 4: We train a multimodal fusion model that takes RGB and depth input by supervising the loss of the network and backpropagate from only the single point in the output layer where a real grasp was attempted.

compare to other approaches in grasping opaque, transparent and specular objects?

Using an overhead ASUS Xtion Pro Live RGB-D sensor, we first collected a set of 75 opaque object images, 75 transparent object images, and 75 specular object images that are from home and office retail stores. All images were collected in varying degrees of clutter and lighting conditions. These objects were distinct from those used in the grasping evaluations below. We resized the images to account for differences between our sensors intrinsic parameters and those of the pre-trained FC-GQCNN model. Second, we randomly divide each type of images into two sets: 65 for training, 10 for evaluations. We also applied spatial augmentation (random rotations and flips) to generate approximately 500 paired training images.

Our networks weights are randomly initialized, and the model is trained to convergence using an Adam optimizer with cross-entropy loss. [10] The network architecture was implemented in Python using Tensorflow and Keras. All training was performed on a machine running Ubuntu 16.04 with an NVIDIA GTX 1080 Ti GPU, a 2.1 GHz Intel Xeon CPU, and 32 GB RAM allocated per job. All robot experiments were performed on a machine with similar specifications. Our object set is shown in Fig. 5. **We use the same test set as [21]**

The state of the art model is named "FC-GQCNN". We refer the RGB-only supervision transfer model as "RGB-ST". Our multimodal fusion is named "RGBD-W". And we also have other two baselines:

- Grayscale RGB ("RGB-G"): Convert the 3-channel RGB input image to a 1-channel grayscale image and rescale the image. The resulting input is then passed directly through the FC-GQCNN network.
- Copy from depth ("RGB-C"). Modify the FC-GQCNN network, which normally receives a one-channel depth input, to receive a 3-channel RGB input by copying the weights of the input layer three times. Rescale the RGB image and then pass it through the modified network.

We use the same test settings as [21]. Each object has 5 real-world grasp attempts. And the results of FC-GQCNN, RGB-C, RGB-G and RGB-ST are all directly from [21].

TABLE I: Performance on individual object grasping

Method	Opaque	Transparent	Specular
FC-GQCNN*	0.733	0.200	0.493
RGB-G*	0.533	0.333	0.413
RGB-C*	0.147	0.240	0.240
RGB-ST***	0.867	0.853	0.640
RGBD-W**	0.587	0.6	0.507

*Trained on simulated grasps **Trained on RGB-D images ***Trained on simulated grasps and opaque object images

Our method RGBD-W performs better than FC-GQCNN, RGB-G and RGB-C but worse than RGB-ST. One of the possible reasons our method did not outperform the RGB-only method could be that our dataset is too small. Many failures were due to noisy predictions as shown in Fig. 7. Calibration of the robot with the RGB-D sensor was another potential source of errors.

Fig. 5: Data collection setup and example images from the dataset of all three types of objects (opaque, transparent, specular).

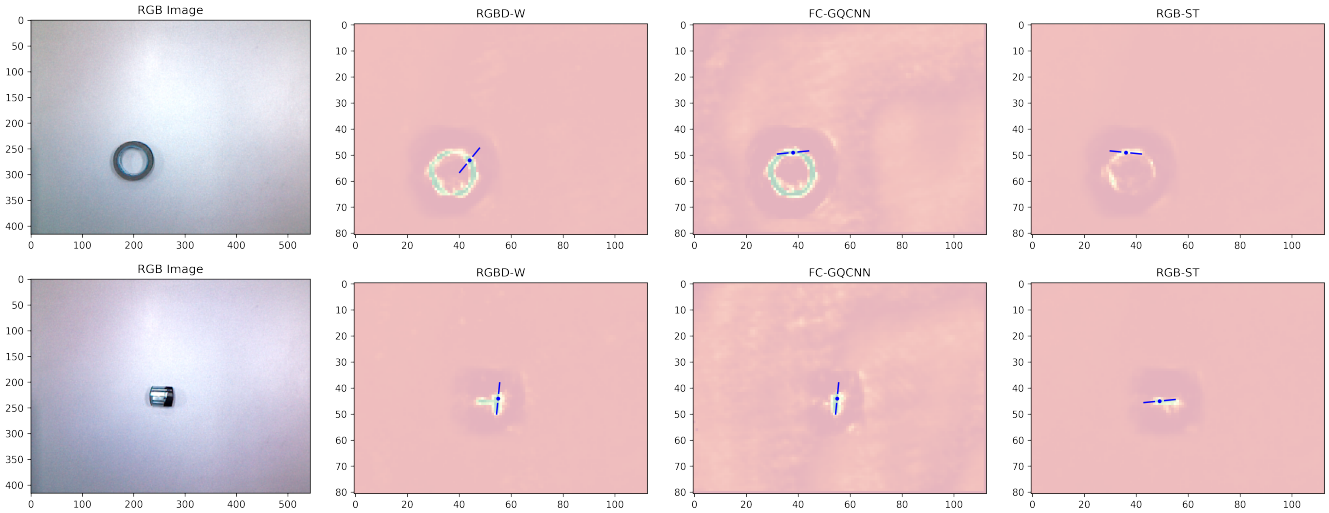


Fig. 6: Some samples for FC-GQCNN, RGB-ST and RGBD-W(Image parameters: alpha=0.3, vmin=0.0, vmax=1.0)

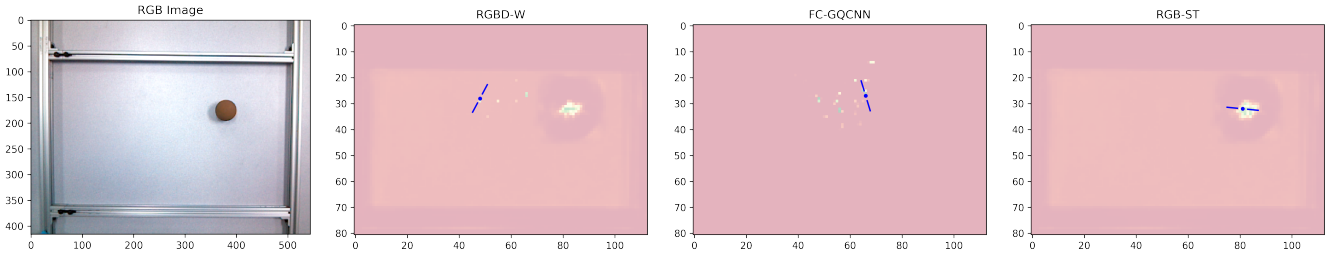


Fig. 7: Due to our relatively small training dataset, many grasp failures were related to the weighting model failing to ignore noisy predictions in the FC-GQCNN output.

V. CONCLUSIONS

We present an adaptive multimodal fusion model for grasping transparent and specular objects. Our model combines depth-based and RGB-based models with supervised learning, and learns directly from real grasps, allowing for online learning. We show our fusion model has better performance than the state of the art depth-only model for transparent and specular objects and similar performance for opaque objects.

For future work, we would like to train our method on a larger dataset to hopefully achieve better performance. We would also like to train and evaluate on different object sets, and evaluate against an existing fusion baseline like Mixture of Deep Experts (MoDE) [13].

ACKNOWLEDGMENT

First and foremost, I would like to show my deepest gratitude to my supervisors, Thomas Weng and Prof. David Held. I shall extend my thanks to Rachel Burcin and Prof. John M. Dolan for all their kindness and help. I would like to thank my parents, Fengchang Tang and Yan Chen, for all their help. Last but not least, I thank Carnegie Mellon University and ShanghaiTech University.

REFERENCES

[1] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000.

[2] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesisa survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[3] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE, 2015.

[4] Markus Enzweiler and Dariu M Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 20(10):2967–2979, 2011.

[5] Carlo Ferrari and John F Canny. Planning optimal grasps. In *ICRA*, volume 3, pages 2290–2295, 1992.

[6] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European conference on computer vision*, pages 345–360. Springer, 2014.

[7] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836, 2016.

[8] Judy Hoffman, Saurabh Gupta, Jian Leong, Sergio Guadarrama, and Trevor Darrell. Cross-modal adaptation for rgb-d detection. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5032–5039. IEEE, 2016.

[9] Ivo Ihrke, Kiriakos N Kutulakos, Hendrik PA Lensch, Marcus Magnor, and Wolfgang Heidrich. Transparent and specular object reconstruction. In *Computer Graphics Forum*, volume 29, pages 2400–2426. Wiley Online Library, 2010.

[10] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[11] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[12] Guanbin Li, Yukang Gan, Hejun Wu, Nong Xiao, and Liang Lin. Cross-modal attentional context learning for rgb-d object detection. *IEEE Transactions on Image Processing*, 28(4):1591–1601, 2018.

[13] Oier Mees, Andreas Eitel, and Wolfram Burgard. Choosing smartly: Adaptive multimodal fusion for object detection in changing environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–156. IEEE, 2016.

[14] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, 1988.

[15] Cristiano Premebida, Joao Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining rgb and dense lidar data. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4112–4117. IEEE, 2014.

[16] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[17] Vishal Satish, Jeffrey Mahler, and Ken Goldberg. On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters*, 4(2):1357–1364, 2019.

[18] Joel Schlosser, Christopher K Chow, and Zsolt Kira. Fusing lidar and images for pedestrian detection using convolutional neural networks. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2198–2205. IEEE, 2016.

[19] Jungwon Seo, Soonkyum Kim, and Vijay Kumar. Planar, bimanual, whole-arm grasping. In *2012 IEEE International Conference on Robotics and Automation*, pages 3271–3277. IEEE, 2012.

[20] Jörg Wagner, Volker Fischer, Michael Herman, and Sven Behnke. Multispectral pedestrian detection using deep fusion convolutional neural networks. In *ESANN*, 2016.

[21] Thomas Weng, Amith Pallankize, Yimin Tang, and David Held Oliver Kroemer. Transfer learning for multi-modal grasping on transparent and specular objects. 2019.

[22] Wikipedia contributors. Stereo camera — Wikipedia, the free encyclopedia, 2019. [Online; accessed 9-August-2019].

[23] Wikipedia contributors. Structured-light 3d scanner — Wikipedia, the free encyclopedia, 2019. [Online; accessed 9-August-2019].

[24] Wikipedia contributors. Time-of-flight camera — Wikipedia, the free encyclopedia, 2019. [Online; accessed 9-August-2019].

Learning Low-Level Continuous Control for Highway Ramp Merging in Dense Traffic

Samuel Triest[1], Adam Villaflor[2] and John M. Dolan[3]

***Abstract*— Several key scenarios, such as intersection navigation, lane changing, and ramp merging, are active areas of research in autonomous driving. In order to properly navigate these scenarios, autonomous vehicles must implicitly negotiate with human drivers. Prior work in driving behaviors presents reinforcement learning as a promising technique, as it can leverage data as well as the underlying decision-making structure of driving with interaction. However, current RL-based approaches to the ramp merging problem in particular rely on hand-tuned reward functions and assume a fixed merge point. In this work, we use Generative Adversarial Imitation Learning (GAIL) to learn a low-level control policy for ramp merging from expert demonstrations. By learning a reward function from expert trajectories, we are able to imitate human behavior while avoiding a hand-designed reward function. Additionally, directly learning acceleration and heading commands allows the policy to initiate the merge at any point. Additionally, we use a masking mechanism based on a distance-keeping model to keep our policy from executing some unsafe behaviors. We validate our approach on a simulation using real-world highway data to demonstrate our algorithm's performance on a variety of driving scenarios.**

I. Introduction

Autonomous driving-related technology has been steadily on the rise since the 2007 DARPA Urban Challenge. Although there exist a number of commercially available advanced driving assistance systems (ADAS) for highway driving in today's vehicles, there remain a number of scenarios that are difficult for ADAS. Such scenarios, which include highway ramp merging, are challenging for ADAS to properly negotiate because of the need to interact with other vehicles. Designing agents that can successfully execute these maneuvers in a wide variety of driving conditions is a major challenge, as the ego-vehicle must be able to interpret the intent of other vehicles and react accordingly. Furthermore, the long adoption window of autonomous vehicles means that autonomous vehicles will share the road with human drivers for some time. This necessitates that autonomous agents be able to interact with drivers in a human-like way.

Given the above challenges, there is recent work in data-driven approaches to designing autonomous vehicle behaviors. Representative data-driven techniques for designing these agents include probabilistic graphical models [1], [2], reinforcement learning [3], [4], and imitation learning via inverse reinforcement learning [5].

This paper extends previous work in ramp-merging behaviors by using inverse reinforcement-based imitation learning to learn low-level control for ramp merging. In order to address potential safety concerns, we use a masking mechanism based on the intelligent driver model (IDM) [3], [6], [7]. Our approach is validated on real-world highway driving data [8], and is able to successfully execute merging behaviors using low-level control outputs.

II. Related Work

Design of effective merging behaviors has been considered by a number of researchers. For the 2007 DARPA Urban Challenge, Urmson et al. [9] use a slot-based approach, where a number of potential merge slots (i.e., between car 1 and car 2, between car 2 and car 3) are identified. The feasibility of each slot is evaluated using a rule-based planner, and the ego-vehicle uses a motion planner to attempt to merge into the best slot.

More recently, a number of data-driven approaches have been considered for the merging problem.

Dong et al. [1], [2], [10] use probabilistic graphical models (PGMs) to estimate the host vehicle's intent to yield. Given the intent to yield from the PGM, the merging vehicle can use traditional control techniques to execute the merge. The design of the PGM relies on observed velocity data from a fixed number of past timesteps, as well as the time-to-arrival to a fixed merge point. By encoding the relationship between these observations and intent to yield in a PGM, the authors are able to predict a given host vehicle's intent to yield to a given merge vehicle [1]. The authors extend this framework to apply to multiple host and merge vehicles and without the assumption of a fixed merge point [2], [10].

Kuefler et al. [5] and Bhattacharyya et al. [11], [12] use imitation learning to generate highway driving behaviors in general. These approaches all rely on using variations of the GAIL algorithm [13] on the NGSIM dataset [8] to construct policies that can replicate the behavior of the original human-driven vehicle. Kuefler et al. consider the single-agent case, while Bhattacharyya et al. extend the work of Kuefler et al. by extending GAIL to the multi-agent case and augmenting the imitation learning with additional hand-tuned constraints. While these authors have shown that imitation learning-based approaches work well when applied to general highway driving (i.e., no particular focus on a given scenario) it is unclear how effective they are on the more difficult merging scenario.

[1]Samuel Triest is an undergraduate in the Computer Science Department at the University of Rochester, Rochester, NY 14627, USA striest@u.rochester.edu

[2,3]Adam Villaflor and John Dolan are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213, USA avillafl, jdolan@andrew.cmu.edu;

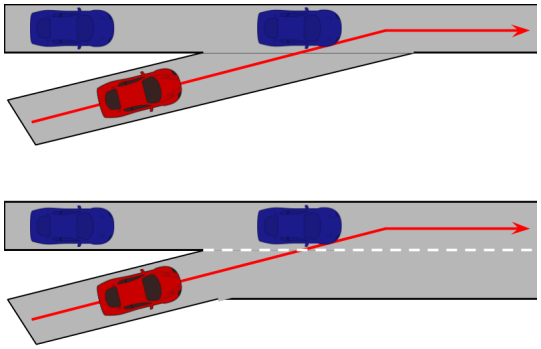


Fig. 1: The merging problem - the ego vehicle (in red) needs to move from the merging lane to the host lane without disrupting the host vehicles (in blue). Two ramp geometries are shown - the top geometry does not contain an auxiliary lane, while the bottom one does.

Hu et al. [3] and Bouton et al. [4] use reinforcement learning to solve the merging problem. Hu et al. employ a multi-agent actor-critic approach. Specifically, they use a decentralized-actor, centralized-critic approach, where actors are limited to their own observations, but the critic is able to observe all actors' observations. Such a model allows for better updates at training time while allowing agents to learn policies that use a realistic set of observations. Bouton et al. employ a single-agent approach that uses an additional PGM to estimate driver intent to yield. The output of the PGM is used as input to the learned policy. Both reinforcement learning-based approaches have the distinct advantage of being able to function end-to-end; the learned policies are able to take observations from sensors as input and output specific driving commands (i.e., acceleration). However, both approaches assume the agent follows a given path to a fixed merge point, where the policy uses its observations to output an acceleration along this given path. As a result of these assumptions, generalizing these approaches to arbitrary ramp geometries is potentially challenging. Furthermore, even in cases with a traditional on-ramp structure (as illustrated in figure 1), there is a significant amount of longitudinal space to execute a merge [14], thus making a path to a fixed merge point potentially limiting.

III. Problem Formulation

We formulate highway merging as a partially observable Markov decision process. Using this formulation, we are able to use reinforcement learning and imitation learning techniques in order to design effective merging behaviors.

A. The Merging Problem

For this paper, we consider the merging problem to be one in which a **merging vehicle** is attempting to enter the lane of a **host vehicle**. The merging problem is illustrated in figure 1. In general, the host vehicle has right-of-way and the merging vehicle is expected to yield to the host vehicle. A behavior that is able to successfully negotiate the merging problem will allow the merging vehicle to safely enter the host vehicle's lane with minimal disruption to the host vehicle. For this work, we will consider the ego-vehicle to be a merging vehicle.

B. Markov Decision Processes

A Markov decision process (MDP), formally defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, d_0)$, is a theoretical framework for describing sequential decision problems. In an MDP, an agent or policy, typically denoted as π, must interact with some environment (characterized by the state space $\mathcal{S}$) by taking some action (selected from the action space $\mathcal{A}$). By doing so, the policy will transition the environment to a new state $s' \epsilon \mathcal{S}$ according to the transition model $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longmapsto [0, 1]$, which outputs a probability distribution over states, given the previous state and action. Given the transition model, the reward R can be calculated as $R(s, a, s')$ where $s, s' \epsilon \mathcal{S}, a \epsilon \mathcal{A}$. The discount factor γ defines the relative value of future rewards to current rewards. d_0 is the initial state distribution of the MDP.

The efficacy of an MDP policy π can be determined through its expected sum of discounted rewards (also called return). Rewards r are multiplied by γ^t where t is the number of timesteps in the future r is accrued. By using discounted sums, return is finite even in MDPs with an infinite horizon. In general, we will consider MDP agents to be stochastic policies. That is, $\pi : \mathcal{S} \times \mathcal{A} \longmapsto [0, 1]$ outputs a probability distribution over $\mathcal{A}$, given a state $s \epsilon \mathcal{S}$. Given this, we can define the expected sum of discounted rewards to be:

$$J(\pi) = \mathop{\mathbb{E}}_{\substack{s_0 \sim d_0, a_t \sim \pi(s_t, \cdot), \\ s_{t+1} \sim P(s_t, a_t, \cdot)}} [\Sigma_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})]$$

Along with MDPs are partially observable MDPs (POMDPs), where the agent is unable to observe the entire state. Agents instead receive an observation o from observation space $\mathcal{O}$ according to the emission probability, which depends on the underlying state ($\mathcal{S} \times \mathcal{O} \longmapsto [0, 1]$). The policy acts according to the observation o, but still transitions the underlying state s, i.e., $s_{t+1} = \mathbb{E}_{o_t \sim \mathcal{O}(s_t), a_t \sim \pi(o_t, \cdot)}[P(s_t, a_t, \cdot)]$.

C. Reinforcement Learning

The general goal of reinforcement learning is to provide a principled way of solving MDPs and POMDPs by parameterizing the policy π according to some parameter vector θ (such a policy is denoted as π_θ). In many cases, this θ refers to the weights of a neural network. Reinforcement learning algorithms attempt to find the parameterization of π that yields the highest return. That is, reinforcement learning algorithms solve:

$$\theta^* = \arg\max_\theta \mathop{\mathbb{E}}_{\substack{s_0 \sim d_0, a_t \sim \pi(s_t, \cdot), \\ s_{t+1} \sim P(s_t, a_t, \cdot)}} [\Sigma_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})]$$

.

Central to many reinforcement learning algorithms are state value, state-action value, and advantage functions

($V^\pi(s)$, $Q^\pi(s,a)$ and $A^\pi(s,a)$, respectively). Many foundational reinforcement learning algorithms, including vanilla policy gradient [15] and actor-critic [16], directly compute the gradient of the return with respect to the policy parameters and perform gradient ascent in order to compute updates to the policy. This gradient takes the form:

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi}[\Sigma_{t=t'}^{T} \nabla_\theta \log \pi(a_t|s_t) Q^\pi(s_t, a_t)] \quad (1)$$

$Q^\pi(s,a)$ can be substituted for $A^\pi(s,a)$ [17] or a Monte-Carlo sample of trajectory returns [15]. In the case of $Q^\pi(s,a)$ or $A^\pi(s,a)$, an additional critic network is often used to estimate $Q^\pi(s,a)$ or $V^\pi(s)$ to yield the family of actor-critic algorithms. [16]–[19]

We refer the reader to [20] as a general reinforcement learning resource.

IV. Methodology

We use an inverse reinforcement learning-based approach using GAIL [13] to generate merging trajectories. We give the ego vehicle access to the kinematic information of its nearest neighbors; the vehicles immediately in front of it in the on-ramp, and the three closest vehicles in the host lane.

A. Trust Region Policy Optimization

In this work, we use Trust Region Policy Optimization (TRPO) [21] to train our policy. We choose to use TRPO because of its theoretical justification to provide monotonic increases to policy performance [21], and because of the existing body of work that uses TRPO effectively in autonomous driving [5], [11], [12].

Since TRPO is a conventional reinforcement learning algorithm, it requires a reward function in order to optimize the policy. While reward functions for the merging problem can be designed heuristically by rewarding good behavior such as successful merging, and penalizing bad behavior such as colliding and going offroad, finding an optimal reward function is challenging. This is especially true, given the higher degree of interaction in merging compared to classical reinforcement learning problems or even regular highway driving.

B. Generative Adversarial Imitation Learning

To avoid the need for a hand-designed reward function, we use Generative Adversarial Imitation Learning (GAIL) [13]. GAIL is an imitation learning algorithm that uses a discriminator network to discriminate between state-action pairs (s_i, a_i) sampled from expert trajectories τ_E and the trajectories τ_θ produced by the current policy π_θ. That is, the discriminator D_ψ (parameterized by ψ) attempts to solve the following:

$$\arg\max_\psi \mathop{\mathbb{E}}_{(s,a)\sim\tau_E}[\log(D_\psi(s,a))] + \mathop{\mathbb{E}}_{(s,a)\sim\tau_\theta}[\log(1 - D_\psi(s,a))]$$

In this example, we consider the objective of the discriminator to be outputting 1 for state-action pairs sampled from τ_E and 0 for state-action pairs sampled from τ_θ. The policy can then be trained using traditional reinforcement learning, using $D_\psi(s,a)$ as a reward signal. That is, the policy gradient update can be taken as:

$$\mathop{\mathbb{E}}_{(s,a)\sim\tau_\theta}[\nabla_\theta \log(\pi_\theta(a|s)) \mathop{\mathbb{E}}_{\tau_\theta}[\log(D_\psi(\overline{s}, \overline{a}|\overline{s} = s, \overline{a} = a))]] - \nabla_\theta H(\pi_\theta)$$

where $\lambda \nabla_\theta H(\pi_\theta)$ is an additional entropy term that encourages randomness in the policy, thus increasing exploration of states.

Since the policy receives reward based on how effectively it "fools" the discriminator, convergence is achieved when the discriminator is no longer able to distinguish between the state-action pairs of the expert and those produced by the policy. Essentially, the discriminator serves as a surrogate reward function, where higher reward is given to state-action pairs that closely resemble state-action pairs from τ_E. Because of this, we can use almost any reinforcement learning algorithm to update our policy π_θ. We choose to use TRPO, per the original GAIL implementation. Ultimately, this allows us to train a policy to learn correct merging behaviors by imitating human merging behaviors.

C. Sensitivity to Discount Factor

We find that the discount factor plays a significant role in policy learning. We find that traditional discount factor of 0.99 produces extremely aggressive and unsafe merging behaviors due to the fairly high frequency of the simulation (10Hz) causing lengthy simulation horizons. To quantify the effect of the discount factor, consider a desirable state s (e.g. a state where the ego-vehicle has successfully moved into the host lane). At time t, the reward for the given state will be $r(s)$. If the ego-vehicle were to arrive at state s at timestep $t + k$, the reward will be $\gamma^k r(s)$. For nearly any k, the discounting as a result of merging at a later, potentially safer timestep will be enough to incentivise the optimal policy to drive aggressively and risk collisions, as the return for reaching ideal states quickly outweighs potential risks. ($0.99^{20} \approx .81$, so a policy will be willing to risk a 19% chance of colliding with another vehicle if it can receive a point of reward $2s$ earlier). To avoid this issue, we set $\gamma \approx 0.9999$, or even 1.0.

D. Masking with IDM

Learning low-level control using reinforcement learning algorithms has the potential risk of the policy producing unsafe actions, especially in states that the imitation learning may not have visited. As such, we implement a masking mechanism for distance-keeping based on IDM [6].

IDM is a distance-keeping mechanism that guarantees collision-free driving to a leading vehicle within a single lane. The IDM controls acceleration of a vehicle by setting the acceleration to the following:

$$a_{IDM} = a_{max}(1 - (\frac{v_\alpha}{v_0})^4 - (\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha})^2)$$

Term	Description	Value	Aggregator
a_{max}	Ego-vehicle max acceleration	$4.6m/s^2$	max
v_0	Speed of free traffic	$12.2m/s$	max
s_0	Minimum desired distance	$0.24m$	min
T	Minimum desired headway	$0.7s$	min
b	Maximum comfortable braking acceleration	$-2.0m/s^2$	from [22]

TABLE I: IDM parameters

$$\text{where } s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{a_{max} b}}$$

$s_\alpha, v_\alpha, \Delta v_\alpha$ are the distance to the leading vehicle, velocity of the ego-vehicle, and velocity difference between the ego-vehicle and leading vehicle, respectively.

Using the acceleration given by the IDM as an upper bound on the acceleration given by the policy, i.e. $\pi_{mask}(s) = \min(\pi(s), a_{IDM})$, we can significantly reduce collisions with the leading vehicle.

Our IDM parameters are obtained via aggregation of the merge scenarios extracted from the NGSIM dataset to yield the values in Table I. We choose the 99th/1st percentile value for max and min respectively. By doing so, our distance keeping mechanism resembles the emergent characteristics of the expert trajectories.

V. Dataset

We validate our algorithm on the NGSIM dataset [8]. NGSIM contains 45 minutes of highway driving at 10Hz for US Highway 101 and Interstate 80 in California. Highway 101 contains five main lanes for highway driving, and an on-ramp that becomes a sixth auxiliary lane. Interstate 80 contains six lanes for highway driving, and an on-ramp that does not become an auxiliary lane. As the geometry of Highway 101 does not necessitate a merge, we consider only the I80 dataset for our validation.

Both datasets contain driving with a high degree of interaction. As such, successful merging necessitates that agents learn a variety of interactions between multiple vehicles near the merge points.

VI. Experiment Setup

We implemented an environment to simulate the on-ramp for Interstate 80, and used it as a basis to implement GAIL. The simulation was based on the SISL NGSIM environment [23], and the reinforcement learning algorithms were extended from the rllab framework [24].

A. Environment

Experiments were conducted by randomly selecting a vehicle in the on-ramp to control using the current policy. The remaining vehicles are played back from NGSIM. Our policy is a fully-connected neural network with hidden layer dimensions $(256, 256, 128)$. At every timestep, the policy takes in observations by querying the current set of vehicles and outputs an acceleration and heading. The ego vehicle is transitioned according to the policy output, and the non-ego-vehicles are transitioned according to the NGSIM dataset.

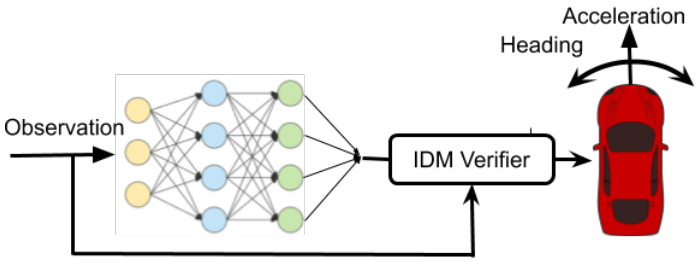


Fig. 2: The ego-vehicle is controlled using a deep neural network that controls the vehicle's heading and acceleration, given an observation. The policy output is verified using a distance-keeping model based on IDM.

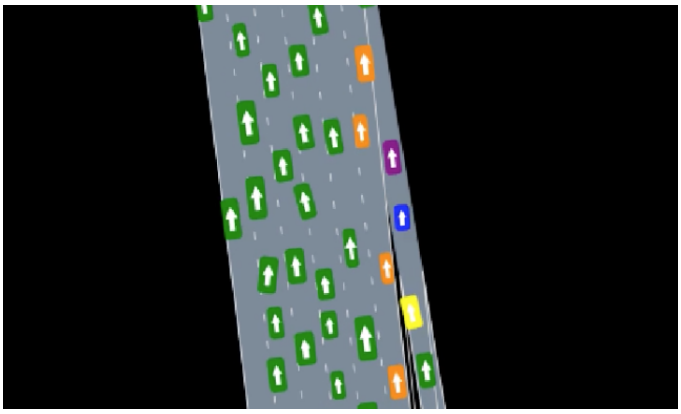


Fig. 3: Diagram of the vehicles that the ego-vehicle has access to - two in the the merging lane and four in the host lane. The ego-vehicle is in blue, the merging neighbors are in yellow, and the host neighbors are in orange. The leading vehicle that IDM is distance-keeping to is in purple. Green vehicles are unobserved by the ego-vehicle.

B. Features

Features for the policy include the kinematic information of the ego-vehicle and its five nearest neighbors: the vehicles immediately in front and behind the ego-vehicle in the merge lane, and the four nearest vehicles by longitudinal distance in the host lane. Higher-order kinematic features are calculated by averaging over a number of previous timesteps. Overall, this amounts to a 93-dimensional feature vector as input to our policy. A complete list of features is provided in Tables II and III.

C. Policy Output

We choose to use a low-level control scheme for our policy by directly outputting the acceleration and heading commands of the ego-vehicle, given its observations. Unlike past approaches, which use some external module to decide the ego-vehicle's trajectory, this approach allows the ego-vehicle more flexibility over its potential merging behaviors.

VII. Results

We validate our results on real-world highway driving data collected from U.S. freeway I-80 [8]. The efficacy of

Feature	Units	Description
Vehicle Length	m	Length of the vehicle
Vehicle Width	m	Width of the vehicle
Lane Offset	m	Lateral dist. from centerline
Lane-relative Heading	rad	Heading from centerline
Lane-relative Speed	m/s	Speed along centerline
Lane-relative Acceleration	m/s^2	Acceleration along centerline
Lane-relative Jerk	m/s^3	Jerk along centerline
Marker Dist Left	m	Distance to the left lane line
Marker Dist Right	m	Distance to the right lane line
Y Displacement to Merge Point	m	Longitudinal dist. to the merge point
X Displacement to Merge Point	m	Lateral dist. to merge point
Merge Completion	Boolean	Indicator of passing merge point
Time To Collision	s	Time until the ego-vehicle collides with the vehicle in front of it, given their current velocities
Is Colliding	Boolean	Indicator of vehicle collision
Off Road	Boolean	Indicator of off-road
Negative Velocity	Boolean	Indicator of driving backward

TABLE II: Ego-vehicle Features

Feature	Units	Description
Vehicle Length	m	Length of the vehicle
Vehicle Width	m	Width of the vehicle
Lane Offset	m	Lateral dist. from centerline
Lane-relative Heading	rad	Heading from centerline
Lane-relative Speed	m/s	Speed along centerline
Lane-relative Acceleration	m/s^2	Accleration along centerline
Lane-relative Jerk	m/s^3	Jerk along centerline
X Displacement to Ego-vehicle	m	Lateral dist. to the ego-vehicle
Y Displacement to Ego-vehicle	m	Longitudinal dist. to the ego-vehicle

TABLE III: Neighboring vehicle Features

our algorithms is measured by playing back the vehicle trajectories collected, selecting a car at the entrance of the highway's on-ramp, and controlling it using our trained policy (with the frequency of control inputs at 10Hz). The simulation ends if the merge is successfully completed, r if the vehicle collides with another vehicle. We observe the collision rate of the ego-vehicle and use it as our primary metric of success.

VIII. Conclusion

This paper presents an effective method for learning ramp merging behaviors via inverse reinforcement learning-based imitation learning. Through a fairly straightforward implementation of GAIL with some additional safety mechanisms via action masking, we are able to achieve reasonable success

Algorithm	Slot-based	iPCB	MML-PGM	GAIL+IDM
Success Rate	85.5%	84.2%	92.4%	90.4%

TABLE IV: Results

in highway merging.

A major limitation of the NGSIM dataset is that non-ego-vehicles are controlled via replay. As a result, vehicles don't properly respond to the trained policy. By incorporating more advanced driving models into our simulation, as well as methods for multi-agent reinforcement learning and data augmentation into our training procedures, we believe that we can get improved results.

IX. Acknowledgements

This work was supported by a grant from the NSF. I (Samuel) would like to thank Carnegie Mellon University, the Robotics Institute and the Robotics Institute Summer Scholars (RISS) program for providing me with a fantastic research opportunity. In particular, I would like to thank Rachel Burcin, Professor John Dolan and Adam Villaflor for their guidance and mentorship this summer. I would also like to thank the other members of Professor Dolan's lab and the 2019 RISS cohort for their support.

References

[1] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1584–1589.

[2] ——, "Interactive ramp merging planning in autonomous driving: Multi-merging leading pgm (mml-pgm)," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[3] Y. Hu, A. Nakhaei, M. Tomizuka, and K. Fujimura, "Interaction-aware decision making with adaptive strategies under merging scenarios," *arXiv preprint arXiv:1904.06025*, 2019.

[4] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," *arXiv preprint arXiv:1906.11021*, 2019.

[5] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 204–211.

[6] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[7] M. Mukadam, A. Cosgun, A. Nakhaei, and K. Fujimura, "Tactical decision making for lane changing with deep reinforcement learning," 2017.

[8] "Next generation simulation (ngsim)." [Online]. Available: https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

[9] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[10] C. Dong, J. M. Dolan, and B. Litkouhi, "Smooth behavioral estimation for ramp merging control in autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1692–1697.

[11] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1534–1539.

[12] R. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. R. Driggs-Campbell, and M. J. Kochenderfer, "Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning," 2019. [Online]. Available: https://arxiv.org/abs/1903.05766

[13] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.
[14] "Washington state highway design manual chapter 1360 interchanges." [Online]. Available: https://www.wsdot.wa.gov/publications/manuals/fulltext/M22-01/1360.pdfpage=15
[15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.
[16] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.
[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.
[18] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.
[19] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," *arXiv preprint arXiv:1205.4839*, 2012.
[20] "Welcome to spinning up in deep rl!" [Online]. Available: https://spinningup.openai.com/en/latest/index.html
[21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.
[22] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.
[23] "Sisl ngsim environment." [Online]. Available: https://github.com/sisl/ngsim˙env
[24] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

Efficient Dynamic Replanning for Risk Aware Graph Search

Simran Virk[1], Sha Yi[2], Sumit Kumar[3], Jaskaran Singh Grover[4], and Katia Sycara[5]

Abstract— **Robots are increasingly being used for missions in dangerous environments where risks may be dynamic or uncertain. In this paper, we introduce an algorithm that plans the safest possible path in an environment with dynamic and uncertain risk. Previous work introduced the algorithm Risk Aware Graph Search (RAGS) that is able to plan a low risk path within uncertain environments. Similar to that approach, this work models uncertainty as stochastic path traversal costs i.e edge costs that are normally distributed. Deterministic costs are revealed for edges once their sources are reached. To adapt this method to an uncertain environment that is also dynamic (due to moving elements or updates from robot sensors) where the expected cost of traversing paths is subject to change, we introduce a principled method to plan a path called dynamic Risk Aware Graph Search (d-RAGS). This method replans by updating a set of candidate paths computed in the first planning phase rather than recomputing the paths from scratch. We show that d-RAGS is able to compute low risk paths more efficiently on various graph sizes when updates come from a two step lookahead from the current position.**

I. Introduction

Search and rescue missions provide an important application area for robots. They have been used in rescue missions to perform various tasks like mapping, rubble removal, structural inspection and in situ medical assessment and intervention. We consider the scenario of non combatant evacuation. In such missions, the goal is to find the safest path out of the environment to evacuate the non combatants. This problem becomes harder if the environment is uncertain and dynamic. For example, this may be due to moving obstacles/adversaries or updates from robot sensors regarding the environment. In order to use multi-agent systems for such a task, we would especially want to be able to incorporate various updates occurring throughout the graph as robot subteams explore the environment.

Previous work [1] introduced Risk Aware Graph Search (RAGS) which plans a low risk path to a goal by modelling it as a path search problem in a graph with uncertain edge weights. They model uncertainty by having edge weights represented by normal distributions of possible edge traversal costs rather than having a deterministic cost. The true cost of traversing an edge is only revealed dynamically once that edge is reached.

This is a useful construct to model the risk in an uncertain environment, especially if we have no model for the behavior of dynamic obstacles. However, it is possible that initial estimates of risk turn out to be inaccurate - based on sensor information from robot subteams, for example. In this case the planned path may need to be updated dynamically. Rather than repeatedly running RAGS to solve this problem, this paper explores the more efficient method of saving information gained in phase 1 (as described earlier) of the first search and updating that as required to avoid having to recompute the entire subset of paths again.

Experiments are conducted on graph sizes of 60,70,80 and 90 nodes, and various edge update frequencies to compare the runtimes, number of algorithm operations (for a platform independent comparison of compute requirements) and path costs (i.e risk) to show that d-RAGS is able to compute low risk paths more efficiently in the presence of dynamic edge weight updates.

II. Related Work

Planning paths through an uncertain environment is a difficult problem. Many fundamental existing path planning algorithms, for example A* [2] do not include any assumptions of stochasticity in the planning process. Uncertainty in planning may be a result of the external environment or of robot sensor and action uncertainty. Methods for dealing with uncertainty in the environment usually must deal with some tradeoff between risk and rewards. For example [3] which solves risk-sensitive MDPs to minimize distance travelled as well and [4] which considers risk associated with entire paths rather than sub-paths in order to maintain optimal viewpoints of the environment while also minimizing risk. When dealing with exclusively positive edge weights, RAGS makes a tradeoff between minimizing distance and risk as well as between paths with lower mean costs and paths with higher means but also higher variances that may be more desirable one true edge costs are sampled. The ability to make this tradeoff relies on RAGS incorporation of dynamically revealed local edge costs into its path selection.

When planning for a dynamic environment, there are two heavily explored classes of methods to deal with this. The first is to plan in a joint time and space configuration so as to include information about the possible trajectories of dynamic elements as is seen in [5] and [6] However, these require significant computation time and so can only handle lower dimensional configuration spaces.

Another way is to treat the environment as static, plan accordingly and then efficiently update the plan when new information is received due to the dynamic nature of the environment as in [7], [8] and [9] These methods are usually more efficient, however they may suffer from some

[1] Simran Virk, Rice University simran.k.virk@rice.edu

[2,3,4,5] Sha Yi, Sumit Kumar, Jaskaran Singh Grover and Prof. Katia Sycara are with Advanced Agent-Robotics Technolgy Lab, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA shayi@andrew.cmu.edu; sumitsk@cmu.edu; jaskarag@andrew.cmu.edu; katia@cs.cmu.edu

overall suboptimality when dealing with dynamic obstacles as compared to the first class of methods as they do not incorporate motion models for the dynamic elements.
This paper implements and tests an algorithm, d-RAGS or dynamic RAGS that builds on RAGS. RAGS consists of two phases. In phase 1, it searches the graph for a subset of paths likely to yield a low cost path to the goal and in phase 2, it follows a path from this subset while incorporating the edge costs obtained by advancing to new nodes in the graph. d-RAGS, in addition to this, efficiently repairs the solution set computed by the first phase of RAGS to deal with dynamic edge weight updates. In this scenario, the efficiency gain allows us to frequently update paths which reduces suboptimality concerns mentioned earlier.

III. Problem Formulation

The problem considered by this paper is very similar to the one considered by RAGS, i.e finding and executing a risk aware path from a given start vertex to a goal vertex in a graph $G = (V, E)$, where the cost of traversing an edge $e \in E$ is drawn from a normal distribution $N(\mu_E, \sigma_E^2)$. Further, because each edge cost is represented by a normal distribution, the cost of a path $P \subset E$, which is the sum of the edge traversal costs, is represented by $N(\mu_P, \sigma_P^2)$ where $\mu_P = \Sigma_P \mu_E$ and $\sigma_P^2 = \Sigma_P \sigma_E^2$.
The cost of an edge is revealed only when the edge is reached, i.e the current node is one of the vertices of the edge. Thus the problem at a given node consists of comparing the available revealed transitions costs and the subsequent set of paths to the goal associated with each of those transitions.
To traverse the path of least risk, each edge transition must select the next vertex $V_{min} \in V_{t+1}$ such that the following probability holds

$$P(c_{V_{min}} < c_X) \geq 0.5 \;\; \forall X \in V_{t+1}/V_{min}$$

In other words, transitioning to the neighbor V_{min} has a higher probability of returning a low cost path than transitioning to any other neighbor. Here the cost c_N of travelling to the goal through the neighbor N must incorporate both the known cost of travelling to the neighbor N and the cost distributions of all the acyclic paths from N to the goal V_g.
In addition to the above described problem, this paper considers the case where, due to some reason, an edge cost distribution may be updated to a new normal distribution $N(\mu'_E, \sigma_E^{2\prime})$. Here, by updating the pruned portion of the graph computed in phase 1 of a previous iteration, we can reduce compute time in updating the planned path. This paper does not consider the cause of the update and assumes that the update reveals the parameters of the expected edge cost, i.e the mean and variance.

IV. RAGS

To solve the above described problem, RAGS follows a two phase approach. First, it computes a set of candidate paths to the goal. Each of these paths is at least a threshold probability i.e d_{thresh} likelier than any other excluded path to yield a low cost of travelling to the goal, where the d_{thresh} is a user defined parameter. In Fig 1, this set is shown in red. In the second phase of the algorithm, it computes the best path step by step based on the local revealed edge costs and the candidate path sets to the goal. In the Fig 1, this path is shown in blue.

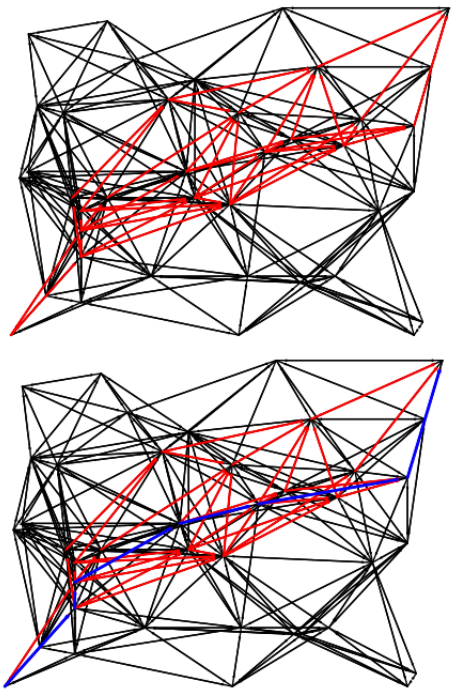


Fig. 1: The above graphs show the set of candidate (non dominated) paths to the goal in red. The final path chosen is shown in blue

A. Phase 1: Non dominated path set

Here a path is considered to 'dominate' another path if it is likelier than some user defined threshold $d_{thresh} \in [0.5, 1)$ that the first path will yield a lower cost to travel to the goal than the second path will. In this phase, the algorithm computes a non-dominated path set to the goal, i.e the set of paths which are more than d_{thresh} likelier to yield a low cost of travelling to the goal than any other path not included in this non dominated set.
As described in [1], a path A is said to dominate a path B, i.e $A \succ B$ if

$$A \succ B \iff P(c_A < c_B) > d_{thresh}$$

We can use the fact that

$$P(c_A < c_B) = 1 - P(c_B - c_A \leq 0)$$

which becomes

$$P(c_A < c_B) = 1 - \left(\frac{\mu_A - \mu_B}{\sqrt{\sigma_A^2 + \sigma_A^2}}\right)$$

where is the cdf of the normal function and path A has mean and variance μ_A, σ_A^2 and path B has mean and variance μ_B, σ_B^2. This equation finally becomes

$$A \succ B \iff \mu_A < \mu_B + \sqrt{2(\sigma_B^2 + \sigma_A^2)} erf^{-1}(1-2d_{thresh}) \quad (1)$$

The inverse error function produces a value between $(-\infty, 0]$ which scales the contribution of the variances in the path domination comparison. As $d_{thresh} \to 1$, the non dominated path set will grow to contain most of the graph as it will need to be almost certain that a path will be more expensive for it to be excluded from the non-dominated set.

B. Phase 2: Quantifying path risk

Considering two neighbors of the current position, nodes C and D, in order to choose one to travel to, we would like to have a pairwise comparison $P(c_{C_{min}} < c_{D_{min}})$ which describes the probability that the lowest cost path from C to the goal will be cheaper than the lowest cost path from D to the goal. This is expanded to

$$P(c_{C_{min}} < c_{D_{min}}) = \int_{-\infty}^{\infty} P(c_{D_{min}} = x).P(c_{C_{min}} < x)dx$$

Again, as shown in Chung et al [1], this can be solved by using

$$P(c_{C_{min}} < x) = 1 - \Pi_{i=1}^{m} \frac{1}{2} erfc(d(A_i))) \quad (2)$$

$$(c_{D_{min}} = x) = \Sigma_{j=1}^{n} \left[\frac{1}{\sqrt{2\pi}\sigma_{B_j}} exp(-d(B_j)^2).\Pi_{k=1_k \neq j}^{n} \frac{1}{2} erfc(d(B_k)) \right] \quad (3)$$

where

$$d(X_i) = \frac{x - c_{X_0} - \mu_{X_i}}{\sqrt{2}\sigma_{X_i}} \quad (4)$$

Here X is a node in the graph, c_{X_0} is the dynamically revealed cost and μ_{X_i} and $\sigma_{X_i}^2$ are the mean and variance of the i^{th} path from X to the goal.

As seen in equation 1, bounding the path set by considering only non dominated paths (from phase 1) is crucial since this pairwise comparison has a complexity of $O(n^2 m)$ where n and m are the sizes of the path sets to the goal from the two nodes under consideration.

C. RAGS algorithm

The RAGS algorithm is shown in Fig 2

V. D-RAGS

Phase 1 of RAGS is similar to A* [2], a fundamental path planning algorithm that computes the shortest path to a single goal by computing the shortest path to intermediate nodes. In RAGS the set $closed$ contains the set of non dominated paths to the goal as well as each intermediate node. This is similar to how A* stores as $g(n)$ for a node n, the optimal cost of reaching that node. In RAGS, since we are computing a path set rather than a single optimal path in phase1, closed holds a set as opposed to a single cost or path as in $g(n)$ in A*. With this point of view, the non dominated set in RAGS is simply $closed(goal)$ As mentioned earlier, efficient replanning methods for A* have already been developed by Likhachev and Stentz in [9] in which they save the g information computed for each node and during replanning and update the nodes that require updation.

The following sections describe the extensions made to RAGS for efficient dynamic replanning.

A. Reversal of the Planning Direction

As shown in Fig 3, the non dominated path set has a large similar portion between replans after taking a step. Reversing the search direction in phase 1 of the RAGS search, i.e the pruning phase would mean that the root and base of the search tree remain the same between replans. (This is as opposed to searching from the current position, where the root of the tree changes, so all of the path means and variances calculated in one plan will no longer be valid in another plan.) So, the phase1 function mentioned in the dRAGS algorithm before performs the same operations as phase1 of RAGS, but it performs the search in the backward direction. This way the information stored in closed and open can be updated effectively. With this direction reversal, the non dominated set will be $closed(start)$.

B. Updation of non dominated path set

When an edge is updated, all the paths containing that edge need to be re-checked for the non Dominance property within closed. If the edge became better, then we may need to remove some other paths from closed and if it became worse, we may need to add in some other paths from closed. In very simple terms, the strategy in d-RAGS is to first delete all the information that could possibly be invalidated by the change - i.e any paths in open or closed that contain the edge; and then, to decide based on the dominance property, which paths should now be added to the open priority queue. Finally, phase1 is rerun to expand the required paths to get the non dominated path set.

The algorithm UpdateEdge performs the necessary updations to the closed and open sets so that phase1 can be rerun until we get all the new non Dominated paths from goal to start (because of the reverse direction in phase1). As mentioned, it deletes all paths containing the updated edge from closed and open. Then, it checks all paths in closed incoming (in reverse) into the source u of the updated edge through the target v of the updated edge and either adds them to closed (lines 10-15) (if there are no paths from goal to u dominating it), ejecting any newly dominated paths from closed and open, or it does not add it to closed if the path is dominated (lines 6- 8). Finally, if this process has resulted in the net removal of a path incoming (in reverse) to u through v, i.e the edge (u, v) got worse and a path was removed, we need to check all paths incoming (in reverse) to u for whether they should now be added to closed and open since they are no longer dominated. (lines 16-21)

C. d-RAGS algorithm

The dynamic rags algorithm is shown in Algorithm 2. Here the functions phase1() and phase2() within the main d-RAGS

Algorithm 1 RISK-AWARE GRAPH SEARCH

// INITIAL SWEEP
Initialize open, closed, $\mathcal{P}_{ND} \leftarrow \emptyset$ ▷ Initialize open and closed sets, and non-dominated path set
$V_s \leftarrow$ start, $V_g \leftarrow$ goal
$N_s \leftarrow \emptyset$, N_s.append(V_s) ▷ Initialize start node
open.push(N_s) ▷ Place the start node in the open set
while open $\neq \emptyset$ **do**
    $N_0 \leftarrow$ open.pop() ▷ Current search node
    closed.push(N_0)
    if N_0.getVertex() = V_g **then**
        $\mathcal{P}_{ND}$.push(N_0) ▷ Non-dominated path to goal found
    else
        $\mathcal{V}_{t+1} \leftarrow$ getNeighbors($N_0, \mathcal{G}$) ▷ Expand the current node
        for V in $\mathcal{V}_{t+1}$ **do** ▷ Assess all neighboring vertices
            $N \leftarrow N_0$.append(V) ▷ Compute child node
            if notAncestor(V, N_0) $\wedge$ nonDom(N, closed) **then**
                open.push(N) ▷ Open set sorted according to dominance
    if $\neg$nonDom(open.top(), $\mathcal{P}_{ND}$) **then** ▷ End search if all nodes in the open set are
        break dominated by the non-dominated path set

// PATH EXECUTION
$\mathcal{G}_{ND} \leftarrow \mathcal{P}_{ND}$ ▷ Directed graph formed by non-dominated path set
$N \leftarrow \emptyset$
$V_0 \leftarrow V_s$
while $V_0 \neq V_g$ **do**
    N.append(V_0)
    $\mathcal{V}_{t+1} \leftarrow$ getNeighbors($N, \mathcal{G}_{ND}$)
    $\mathcal{V}_{ordered} \leftarrow$ ComparePathSets($\mathcal{V}_{t+1}, \mathcal{G}_{ND}$) ▷ Total ordering of vertices from Equation (3)
    $V_0 = \mathcal{V}_{ordered}$.pop() ▷ Execute edge traversal

Fig. 2: The RAGS algorithm

algorithm correspond to phase1 and phase2 of the RAGS algorithm. However, phase1() now searches backwards with the start as the goal rather than the current position. Further, phase2() no longer steps until the goal is reached, but takes a single step towards the goal and returns the node it travels to. The while loop allows us to check for updates after each step, and if seen, we run the UpdateEdge algorithm for every updated edge followed by another run of phase1 to add the required paths to the non dominated set.

VI. EXPERIMENTS

To compare the performance of RAGS and dRAGS, graphs of various sizes, i.e number of nodes, were generated with position coordinates. The nodes were connected by an edge if the distance between them was below a user defined distance parameter. The variance of each edge was chosen as a random integer between 1 and 10, and the mean of each edge was the sum of the distance between the two nodes and a random integer from 0 to 10.

Algorithm 2: d-RAGS

Input: Graph G, start, goal
Initialize open, closed, P_{ND}
initialPath ← [goal]
open.push(initialPath)
cur ← start
Phase1()
while *cur* $\neq$ *goal* **do**
    if *seenUpdates* **then**
        for *edge in updates* **do**
            UpdateEdge(edge.source, edge.target)
        Phase1()
    cur ← $Phase2()$

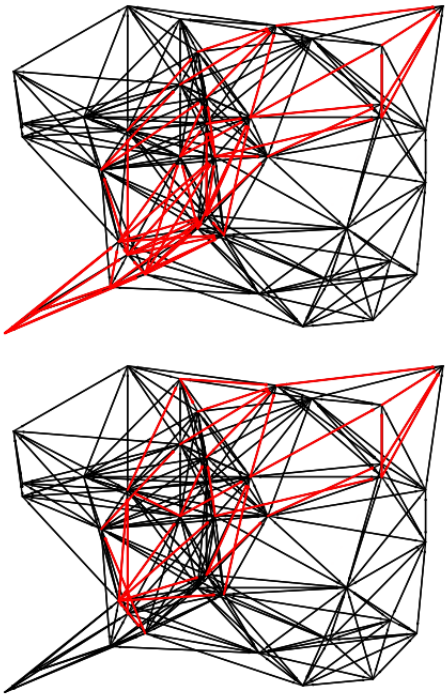


Fig. 3: Much of the searched portion (red) remains the same between replans. In this case, it can improve efficiency to update stored search information rather than recomputing it from scratch

Algorithm 3: UpdateEdge

Input: u, v

```
1  DeleteFromClosed(u,v)
2  DeleteFromOpen(u,v)
3  removals ← False
4  foreach path in closed[v] do
5      newpath ← path + (u,v)
6      if not nonDom(closed, newpath) then
7          if newpath in old(closed) then
8              removals ← True
9      else
10         P_d ←paths in closed dominated by newpath
11         edges ← last edge for p in P_d
12         for edge in edges
13             DeleteFromClosed(edge.source, edge.target)
14             DeleteFromOpen(edge.source, edge.target)
           closed.push(newpath)
15         open.push(newpath)
16 if removals then
17     P
         ← paths incoming to successors(u) in closed
18     for p in P do
19         if p not in closed and nonDom(p, closed) then
20             closed.push(p)
21             open.push(p)
```

We used a set of 20 graphs for each size and varied the edge updation frequency. In these trials, edge updates came from upto a 2 step lookahead, i.e edges emanating from a neighbor of a neighbor of the current node. We used two edge updation frequencies, in which one in 5 or one in 20 of the edges of the graph would be updated from the initial normal distribution if they fell within the range of updation lookahead.

To compare the performance in a platform independent manner, we compared the number of algorithm operations performed by both RAGS and d-RAGS and plotted the percentage of extra operations performed by RAGS. Here, we plot path expansions and heap operations. Path expansions correspond to line x of the d-RAGS algorithm. They represent the scenario where we find that a path to an intermediate node is non dominated and continue to follow the path. These plots are shown in Fig 4
As we can see, the performance gain is more pronounced for larger graphs, and in this case for the lower frequency of edge changes - this is because a very high edge change frequency means that very little information remains the same between runs. This is because the path set becomes more expensive to recompute. Further, heap operations is a better metric as it is able to account for the extra updation work that d-RAGS performs, and so does not provide too optimistic an estimate of d-RAGS's performance.

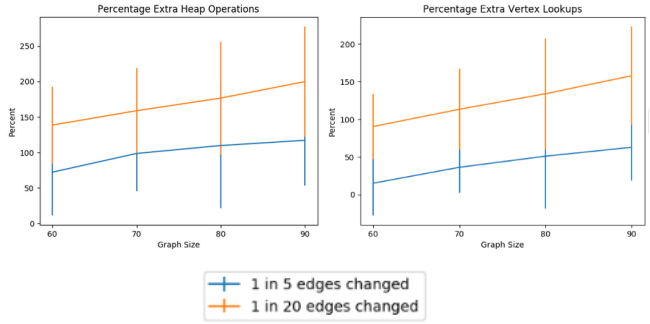


Fig. 4: The above graphs show the percentage of extra algorithm operations that RAGS performs over d-RAGS when replanning for various graph sizes and edge update frequencies. The legend applies to both graphs. For both frequencies, we see that benefit is more pronounced as graph size increases

The difference in computation time and cost for d-RAGS and RAGS is show in Fig 5. These trials were run on an Hp Pavilion laptop with 7.5 GiB ram, 8 cores and intel i5 processor running ubuntu 18.04.

From the above plots and experiments, we find that d-RAGS is able to produce paths of comparable cost to RAGS with much fewer operations.

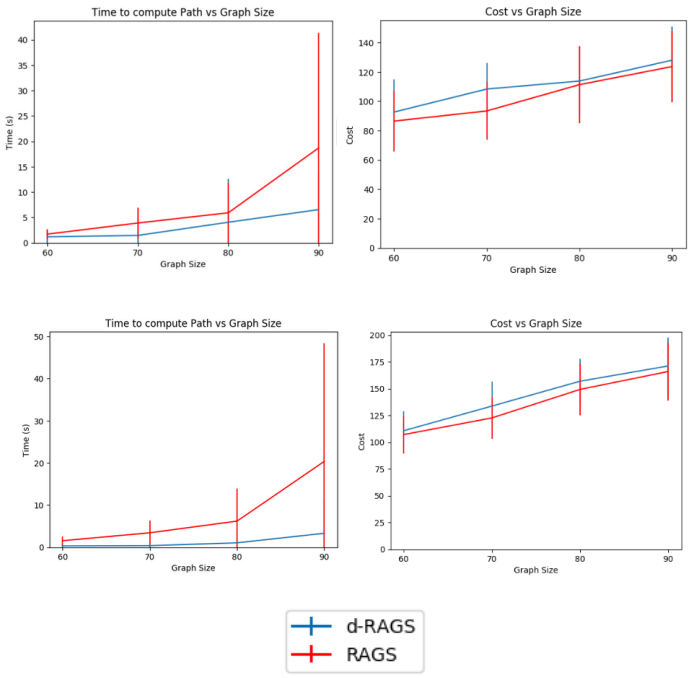


Fig. 5: The graphs above show comparisons of compute time and cost for RAGS and d-RAGS. The first two graphs are for edge update frequency of one in 5 and the lower two graphs are for the frequency of one in 20

VII. Conclusions

Through our experiments we have shown that d-RAGS is able to find a low risk path through a graph with fewer algorithm operations than RAGS in the case of updates coming from 2 steps away from the current position. A good extension of this work would be to further optimize it so that it is efficient in the case of updates coming from anywhere in the graph. This would be useful in a multi agent scenario where a subteam is scouting ahead and updating risk estimates.

This could be achieved by reacting to edge updates more selectively. For example, reacting to an edge update only if the magnitude of the update would produce significant changes in the solution and if the edge update lies in a region of the graph that we are likely to traverse in travelling to the graph. We would not, for example want to react to changes coming from distant or previously traversed areas of the graph, which we would be likely to receive in a multi-agent rescue scenario where sub-teams are exploring different areas of the graph.

Further, as mentioned in [1], this strategy of updating a solution efficiently would also be applicable to the problem of planning a path for information gain. Here edge weight distributions would represent expected information gain from traversing the edge and these would change as the graph is traversed due to submodularity of the data. Information theoretic concepts could be applied here to find the changes in expected information gain and they could be efficiently accounted for using updating of the path using the methods described in this paper.

VIII. Acknowledgement

Along with all the authors, I would like to thank the lab members of the Advanced Agent-Robotics Technology Lab for their support. I would also like to thank Prof. Katia Sycara for giving me the opportunity to work in her lab.

References

[1] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, "Risk-aware graph search with dynamic edge cost discovery," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 182–195, 2019.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "łucorrectionł/u to "a formal basis for the heuristic determination of minimum cost paths"," *SIGART Bull.*, no. 37, pp. 28–29, Dec. 1972. [Online]. Available: http://doi.acm.org/10.1145/1056777.1056779

[3] S. Carpin, Y.-L. Chow, and M. Pavone, "Risk aversion in finite markov decision processes using total cost criteria and average value at risk," 02 2016.

[4] X. Xiao, J. Dufek, and R. Murphy, "Explicit-risk-aware path planning with reward maximization," 2019.

[5] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kino-dynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[6] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 2210–2215.

[7] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.

[8] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, pp. 968–975 vol.1, 2002.

[9] S. Koenig and M. Likhachev, "D* lite," in *Eighteenth National Conference on Artificial Intelligence*, 2002.

Trajectory Tracking for Aggressive Quadrotor Flights Using $\mathcal{L}1$-Adaptive Controller

Sushant Uttam Wadavkar[1], Ramkumar Natarajan[2] and Maxim Likhachev[3]

***Abstract*— Trajectory optimization techniques are used to exploit large computational dimensions and complex systems, but these techniques result in locally optimal solutions. On the other hand, discrete graph search techniques which provide bounded suboptimality guarantees, suffer the curse of dimensionality. The success of this research would pave the way to reasoning complex planning problems involving high dimensional dynamical systems like aggressive quadrotor flights which give access to their competitive capabilities. Despite the many applications of quadrotors, their full potential, especially in terms of autonomy and agility, has not been exploited yet. Making quadrotors more autonomous and more agile brings the benefit of requiring fewer operators and completing tasks faster, which makes them more useful and increases their profitability. However, making quadrotor platforms more agile, e.g. by making them smaller, also makes them more difficult to control due to faster dynamics. Such applications require precise control mechanism to track those complex plans (or trajectories). This study focuses on controller synthesis for one representative and other significant domains (aggressive quadrotor flight). Since this environment involves smaller ultimate bounds on tracking errors and variable uncertainties, we have come up with the controller which provides robust performance and stabilization for dynamic uncertain disturbances. The method that comes close to achieving this is H∞ loop-shaping. Along with the robustness we also aim to demonstrate the $\mathcal{L}1$- Adaptive Control strategy which makes the system adaptive to unmodelled disturbances. $\mathcal{L}1$- Adaptive Control tries to compensate for the uncertainties within the control bandwidth of the actuator. The developed control strategy is benchmarked against standard techniques in the literature for quadrotor control like PID and MPC. In this paper we present such a scheme that guarantees good transient performance and robustness.**

I. INTRODUCTION

Currently, the quadrotors are being actively experimented upon instead of helicopters or other aerial vehicles. This can be easily reason based on the facts that the kinetics energy required for UAV flight rotors is much lesser than other aerial vehicles, the landing and taking off of quadrotors can happen in compact places and safely, no additional mechanism is needed for changing the blades pitch during their flight, also quadrotors provide several applications in defense and industrial fields. In this work, we are especially focusing on quadrotor flights in aggressive maneuvers and cluttered environments. Quadrotor flights for such environments can closely be related to real-life environments.

[1]Sushant Uttam Wadavkar is a senior student of Mechanical Engineering Department at Indian Institute of Technology Madras, Chennai-600036, India me16b172@smail.iitm.ac.in

[2,3]Ramkumar Natarajan, Maxim Likhachev are with Search Based Planning Laboratory, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA rnataraj@andrew.cmu.edu; maxim@cs.cmu.edu

In this research work, we are considering four-rotor aircraft for flight control. Quadrotor has six degrees of freedom and has four input parameters viz. Three rotational moments and one parameter for thrust value. UAVs used are underactuated and non-linear coupled systems that raise the problem of its flight control [1]. Many control system models have been proposed over last two decades. Earlier in the case of other aerial vehicles, linear control systems have been successfully developed [2] [3] [4]. The area of non-linear control systems has recently started expanding capabilities of quadrotors. The non-linear control scheme focuses on three main techniques: sliding mode control, backstepping control, and adaptive control scheme.

Unlike this research work, many models eliminate the disturbances and the uncertainties for simplicity viz. PID [5], sliding mode [6] or robust controllers [7]. A backstepping control method is proposed in [8] by considering an aggressive perturbation with bounded signals. This approach of sliding mode controller causes chattering problems that may excite high-frequency unmodeled dynamics. This results in certain limitations on handling uncertainties. Widely, PID controllers are adopted, but it is required that uncertainties are fixed. Because of the limitations mentioned above, adaptive controllers have been developed and are very popular for tracking the trajectories in existence of disturbances and uncertainties. In this study, we are considering the $\mathcal{L}1$-AC [9] which is an ad hoc modification of a standard Model Reference Adaptive Control (MRAC) for plants whose full state is available by inserting a low pass filter at its input.

Due to the limitations mentioned above, adaptive controllers have been developed and are very popular for tracking desired trajectories in existence of disturbances and uncertainties [10] [11] [12] [13]. Although adaptive controllers are very robust for quadrotors trajectory tracking in existence of disturbances and uncertainties, most of the studies are based on linearization [10] [11] or simplification [12] [13]. In [12] only the constant external disturbances is considered into the system dynamics and the stability analysis. In [13] an adaptive block backstepping controller is presented to stabilize the attitude of a quadrotor, however this method only guarantees the boundedness of errors.

II. MODELING AND SYSTEM IDENTIFICATION

A. Quadrotor - System Description

The autonomous aerial vehicle used in this paper is a four rotor helicopter. The quadrotor is propelled by four rotors. The motion of this vehicle is controlled by changing the speed of rotation of the four rotors. The quadrotor

is a typical under-actuated, non-linear coupled system as shown in Fig.(1). In order to obtain forward motion, the speed of rotation of the rear rotor must be increased and, simultaneously, the front rotor velocity must be decreased. The lateral motion is achieved with the same strategy but using the right and left motors. Yaw rotation is results from the difference in the counter-torque between each pair of propellers, i.e., accelerating the two clockwise turning rotors while decelerating the counterclockwise turning rotors, and vice-versa.

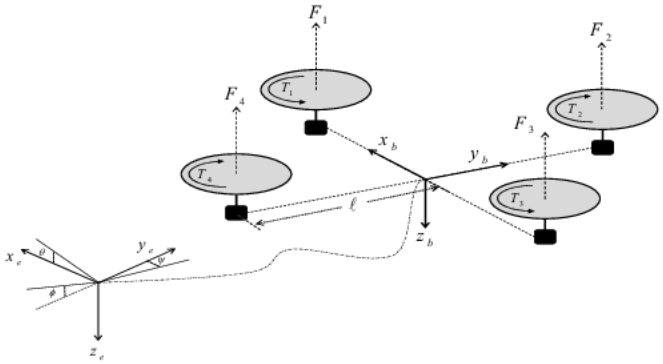


Fig. 1: Coordinate systems and forces/moments acting on the quadrotor

The acting forces and moments applied onto a rotating propeller can be described as shown in Fig.(1). The object coordinates system is related to the translational positions (x, y, z) and the attitude described by three angular Cartesian axes (θ, ϕ, ψ), respectively. We assume that the propeller j^{th} rotates in the **XY** plane, the rotating speed is ω_j, the propeller is moving sideways through the air with velocity **V**, and the propeller rotates about a particular axis of angular velocity α. Thrust force (upward) is the result of drag torque Q_j (inverse direction with ω_j) on the rotor shaft, which is generated by the power applied to each motor. Sideways movement with velocity **V** generates hub force H_j (inverse direction with **V**) and hub moment R_j (unbalanced lift between advancing and retreating blades). The gyroscopic effect of a rotary propeller produces a gyroscopic moment G_j corresponding to angular velocity Ω.

The dynamic model of the system is obtained under several reasonable assumptions. First, the vehicle is a rigid body in space and therefore, Newton-Euler equations can be used to describe its dynamics. Second, by reason of linear velocities of the quadrotor $(\dot{x}, \dot{y}, \dot{z})$ are relatively low, the effects of hub force and moment are assumed to be negligible. Third, the quadrotor helicopter is symmetrical with respect to the x, y, and z axes

B. Quadrotor Kinematics Model

Let $\{E\} = \{X_e Y_e Z_e\}$ denote an earth-fixed inertial frame of reference and $\{B\} = \{XYZ\}$ be the body-frame whose origin coincides with the center of mass of quadrotor. Let the absolute position of Quadrotor be $P = (x, y, z)$ and the attitude be Euler angle $\Theta = (\phi, \theta, \psi)$. Using a rotation matrix $\boldsymbol{R}$, airframe orientation in space can be achieved from inertial frame to body frame, where $R \in SO(3)$ is an orthogonal matrix

Model Equations: As shown in Fig.(1), $\boldsymbol{F_i}$s are the forces over each propeller. Thee equivalent force on the quadrotor can be decoupled into XYZ directions. Let equivalent forces in X, Y, Z be $F_x, F_y, and F_z$ respectively in the body axis. So, $\boldsymbol{F} = [F_x, F_y, F_z]^T$. And this $\boldsymbol{F}$ can be expressed as

$$\boldsymbol{F} = m(\dot{\boldsymbol{V}} + \boldsymbol{\Omega} \times \boldsymbol{V})$$

V be the velocity of quadrotor in the body axis, can be given as $\mathbf{V} = [u, v, w]^T$ and Ω is the rotational velocity in the body reference frame, which is expressed as $\boldsymbol{\Omega} = [p, q, r]^T$. m denotes the quadrotor mass. **V** can be expressed as

$$\dot{\boldsymbol{V}}_{\boldsymbol{t}} = (\frac{\boldsymbol{F}}{m} - \boldsymbol{\Omega} \times \boldsymbol{V})$$

where $\mathbf{V}_t$ denotes partial derivative of velocity vector with respect to time. For the rotational acceleration

$$\boldsymbol{M} = (I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega})$$

where **M** is the vector of the moment in the body axis, **I** is the inertia matrix, a symmetric positive definite constant matrix express in frame {B}, and $\dot{\boldsymbol{\Omega}}$ is the angular rotation acceleration. The transformation from wind-axis to body-axis is defined as

$$\begin{bmatrix} u & v & w \end{bmatrix}^T = \boldsymbol{V} \begin{bmatrix} c(\alpha)c(\beta) & s(\beta) & s(\alpha)c(\beta) \end{bmatrix}^T$$

where s(.)=sin(.), c(.)=cos(.). Also for inverse transformation

$$V = \left|\mathbf{V}\right| = \sqrt{u^2 + v^2 + w^2}$$

and α and β can be given as $\alpha = \tan^{-1}(\frac{w}{V})$; $\beta = \sin^{-1}(\frac{v}{V})$.

Upon simplifying

$$\dot{V} = \frac{1}{m} \begin{bmatrix} -D \\ Y \\ F_{xb} \\ F_{yb} \\ F_{zb} \end{bmatrix}^T \begin{bmatrix} c\beta \\ s\beta \\ c\alpha c\beta \\ s\beta \\ s\alpha c\beta \end{bmatrix}$$

$$-mg(c\alpha c\beta s\theta - s\beta s\phi c\theta - s\alpha c\beta c\phi c\theta)$$

$$\dot{\alpha} = \frac{1}{V * m * c\beta} \begin{bmatrix} -L \\ F_{zb} \\ F_{xb} \\ mg \end{bmatrix}^T \begin{bmatrix} 1 \\ c\alpha \\ s\alpha \\ c\alpha c\phi c\theta + s\alpha s\theta \end{bmatrix}$$

$$+q - \tan\beta(p * c\alpha + r * s\alpha)$$

$$\dot{\beta} = \frac{1}{V * m} \begin{bmatrix} D \\ Y \\ -F_{xb} \\ F_{yb} \\ -F_{zb} \end{bmatrix}^T \begin{bmatrix} s\beta \\ c\beta \\ c\alpha s\beta \\ c\beta \\ s\alpha s\beta \end{bmatrix}$$

$$+mg(c\alpha s\beta s\theta + c\beta s\phi c\theta - s\alpha s\beta c\phi c\theta) + p * s\alpha - r * c\alpha$$

Where D is the aerodynamic drag, Y is the lateral side force, L is the aero- dynamic lift and g the gravity acceleration. Then F_{xb}, F_{yb}, F_{zb} are the propulsive forces in body

axis. The equations regarding $\dot{\Omega}$ that are part of the state are more precisely, with the assumption that the matrix of inertia I is diagonal, the following:

$$\dot{p} = \frac{l + (I_y - I_z)qr}{I_x}$$

$$\dot{q} = \frac{M + (I_x - I_z)pr}{I_y}$$

$$\dot{r} = \frac{N + (I_x - I_y)pq}{I_z}$$

The attitude in Euler's angles can be expressed as

$$\boldsymbol{\Omega} = \boldsymbol{R}\dot{\boldsymbol{\Theta}}$$

R is the matrix that transform angular velocities in earth-fixed axis system into body axis angular velocities. And **R** can be given as

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix}$$

Upon simplifying we get

$$\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix} = \begin{bmatrix} p & q & r \end{bmatrix} \boldsymbol{W}$$

where $\boldsymbol{W}$ matrix is

$$\boldsymbol{W} = \begin{bmatrix} 1 & 0 & 0 \\ s\phi \tan\theta & c\phi & s\phi \sec\theta \\ c\phi tan\theta & -s\phi & c\phi sec\theta \end{bmatrix}$$

and $det(\boldsymbol{W}^T) = sec\theta$. Thus, the matrix $\boldsymbol{W}$ is invertible when the pitch angle satisfies $\theta \neq \pm(2k+1)\frac{\pi}{2}, k \in \mathbb{Z}$. The velocities in the earth-fixed coordinate $(\dot{x}, \dot{y}, \dot{z})$ are related to corresponding velocities in body-axis velocities **V** by

$$\begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \mathbf{L}_{BE}\mathbf{V}$$

where $\mathbf{L}_{BE}$ can be given as

$$\begin{bmatrix} \mathbf{L}_{BE} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{bmatrix}$$

Thus we get velocity values for quadrotor in the earth coordinate axis viz. $\begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$.

III. Controller Design Methodology

The control problem presented in this work is to perform approaching position-attitude tracking and attitude stabilization of the quadrotor helicopter by trajectory tracking controllers based on an L1-adaptive controller. Based on the dynamical model, the control system is divided into two subsystems: position subsystem and attitude subsystem. The whole state equations for quadcopter model developed in the previous system can be rewritten as the following. $\mathbf{x} = [u, v, w, p, q, r, \phi, \theta, \psi, x, y, z]^T$

We consider the following class of plants considered by $\mathfrak{L}$1-Adaptive Controller authors [14] [15]

$$\dot{x}(t) = Ax(t) + B_m u + f$$

The above expression is a reference system where $x \in \mathbb{R}^n$ is the state vector which is assumed to be measurable, $u \in \mathbb{R}$ is the control input, f contains all the modelling uncertainties due to unmodeled dynamics, external disturbances and commands cross-coupling. The above system can be represented as:

$$\dot{x}(t) = A_m x(t) + B_m(\omega u_{ad} + \sigma_1) + B_{um}\sigma_2$$

and the total control law is $u = -K_{baseline}x + u_{ad}$, with $A_m = A - B_m K_{baseline}$, where B_m is the control channel matrix, σ_1 also known as matched disturbance, σ_2 known as unmatched disturbance, and ω is the matrix that represents the cross-coupling among different control input.

It is important to notice that the $\mathfrak{L}$1-Adaptive Controller unlike other adaptive controllers such as MRAC, compensates only the uncertainties within the controller bandwidth by a low-pass filter $D(s)$.

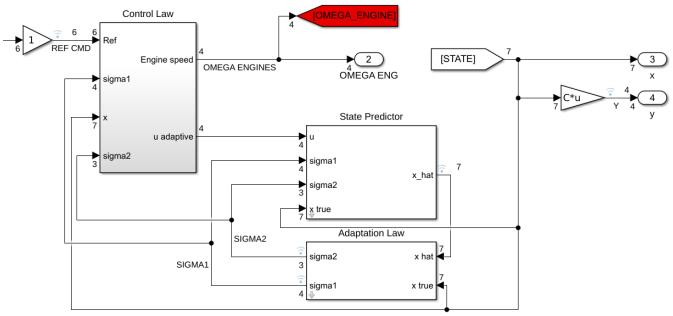


Fig. 2: Block Diagram of L1-Adaptive Controller

A. State Predictor

As shown in the Fig.(2), controller is associated with state predictor which outputs untime estimated values of x. The state predictor equation can be derived from system model representation as follows

$$\dot{\hat{x}} = A_m\hat{x} + B_m(\omega u_{ad} + \hat{\sigma_1}) + B_{um}\hat{\sigma_2}$$

where $\hat{x}$, $\hat{\sigma_1}$ and $\hat{\sigma_2}$ denote the estimated values of states, matched and unmatched disturbances respectively.

B. Adaptation Law

The adaptation laws are used to estimate the matched and unmatched uncertainties. Estimated values of state are then fed to adaptation block to find the error $(\tilde{x})$ with respect to the true values of states. We then get,

$$\tilde{x} = \hat{x} - x$$

$$\mu(T_s) = e^{A_m T_s}\tilde{x}$$

$$\phi(T_s) = A_m^{-1}(e^{A_m T_s} - I_n)$$

$$\begin{bmatrix} \hat{\sigma_1} \\ \hat{\sigma_2} \end{bmatrix} = -\begin{bmatrix} I_m & \dots \\ \dots & I_{n-m} \end{bmatrix} \begin{bmatrix} B_m & B_{um} \end{bmatrix}^{-1} \phi(T_s)^{-1}\mu(T_s)$$

T_s is the sampling time. From equations above, we get desired estimates of uncertainties.

C. Control Laws

As mentioned earlier, $u(t) = -K_{baseline}x + u_{ad}$, where gain value is such $A_m = A - BK_{baseline}$ is Hurwitz. The adaptive control law can be given as

$$u_{ad} = -K_{ad}D(s)\hat{\eta}$$

$$\eta = \omega_o u_{ad} + \hat{\eta_1} + \hat{\eta_2} - r_g$$

$$\hat{\eta_1} = \hat{\sigma_1}$$

$$\hat{\eta_{2m}}(s) = H_m^{-1}(s)H_{um}(s)\hat{\sigma_2}(s)$$

$$r_g = K_g(s)r(s)$$

Where $D(s)$ and K_{ad} are the tuning parameters that have to be set such that

$$C(s) = \omega_u K_{ad}D(s)(I_m + \omega_u K_{ad}D(s))^{-1}$$

For all possible ω_u is stable and has the gain value $C(0) = I_m$; also $C(s)H_m^{-1}$ has to be a strictly proper transfer function. The matrices H_m and H_{um} are defined as

$$H_m(s) = C(sI_n - A_m)^{-1}B_m$$

$$H_{um}(s) = C(sI_n - A_m)^{-1}B_{um}$$

The value of feedforward prefilter K_g is chosen such that it decouples the signals, such that $M(s) = C(sI_n - A_m)^{-1}B_mK_g$ has off-diagonal elements zero DC gain and diagonal gain one

$$K_g = -(CA_m^{-1}B_m)^{-1}$$

TABLE I: Quadrotor Model Parameter

Parameters	m	l	I_x	I_y	I_z	K_t	K_f
Value	0.300	0.5	0.3352	0.3352	0.2943	0.052	0.11
Units	kg	m	$kg.m^2$	$kg.m^2$	$kg.m^2$	$\frac{N.m}{V^2}$	$\frac{N}{V}$

In the presence of unmodeled dynamics $\mathfrak{L}1$-Adaptive Controller above cannot be proven to have globally bounded solutions unless the adaptive gain has a reasonable size and further modifications in the adaptive law are imposed. The negative effect of high adaptive gains on robustness is well documented in the literature of adaptive control [16]. The negative effect of the input filter on robustness is shown in [17]. In the following sections we present a scheme that meets the transient performance and robustness criteria.

IV. Results

An artificial trajectory was fed to the system which replicated aggressive manure quadrotor flight. The 3D plot for that trajectory is shown in Fig.(3).

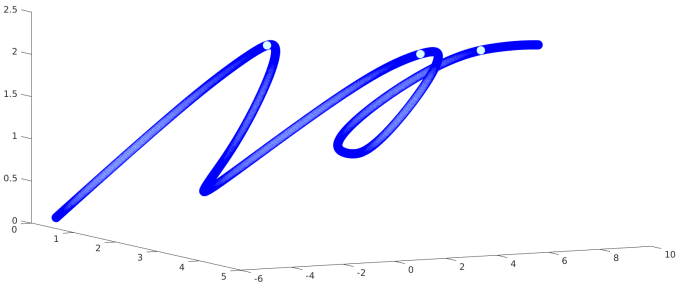


Fig. 3: Sample of aggressive trajectory

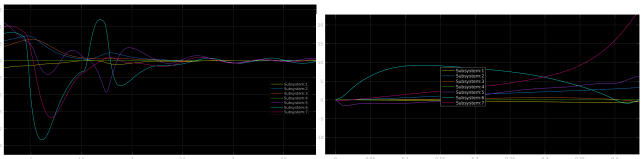
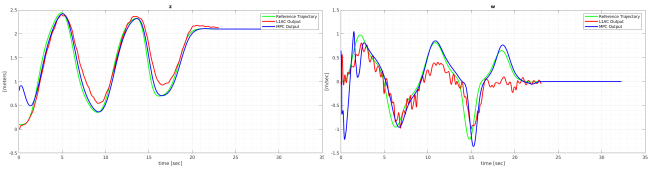
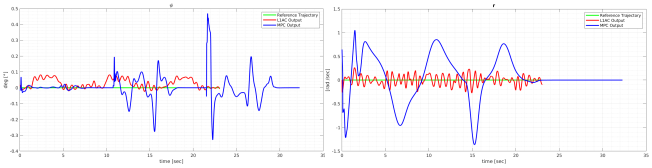


Fig. 4: Comparison of control analysis around a constant state (figure on left) and around aggressive flight (figure on right)



(a) Position in **E** frame (b) Velocity in **B** frame

Fig. 5: Position and velocity



(a) Attitude in **E** frame (b) Angular velocity in **B** frame

Fig. 6: Attitude and angular velocity

V. Numerical Study

System model is updated to contain bounded unmodeled disturbance. The state predictor works in conjunction with the measured states to estimate the state uncertainty. The adaptation law uses the state uncertainty to update the system and control parameters. Finally, the control law uses the updated parameters to synthesize the control input for the uncertain system.

- The time step used in this experiment was 0.01 seconds.
- Larger adaptive gains (of the order $K = 10^6$) lead to numerical instability.
- In the absence of projection, all signals in the closed-loop system go unbounded.

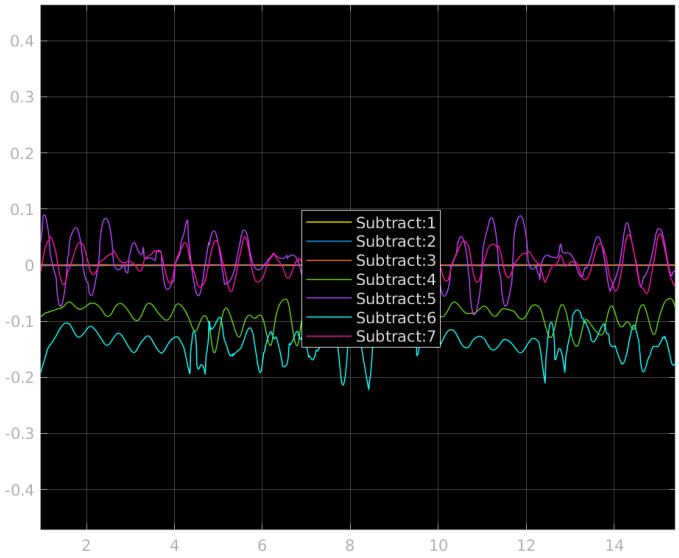


Fig. 7: Error variation for input state values

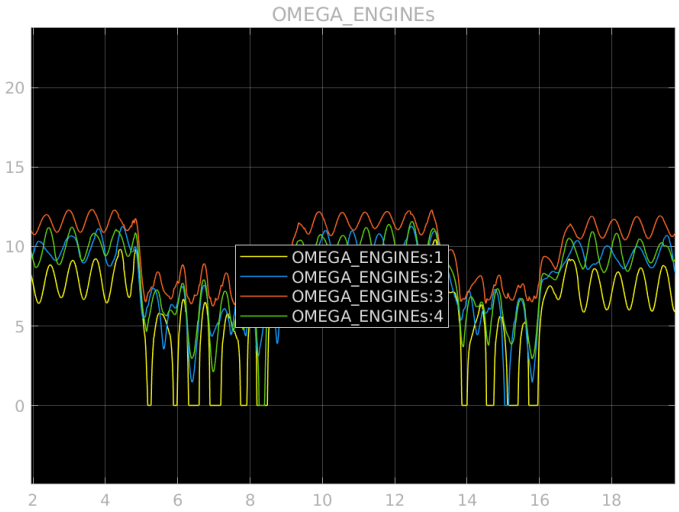


Fig. 8: Angular velocities of quadrotor motors

- Non-zero steady state tracking error remains but the high frequency oscillations are gone due to smaller step size.

VI. Conclusions

A new nonlinear adaptive control system named as $\mathfrak{L}1$-Adaptive Controller is used for trajectory tracking control of quadrotor unmanned aerial vehicles instead of control over stabilization around a point. Controller guarantees that in presence of bounded uncertainties the system remains stable. As we have seen, $\mathfrak{L}1$-Adaptive Controller tries to compensate for the uncertainties within the control bandwidth of the actuator. The given controller, $\mathfrak{L}1$-AC guarantees good transient performance and robustness. It is developed directly on the special orthogonal group to avoid complexities and ambiguities that are associated with Euler-angles or quaternions, and the proposed adaptive control term guarantees almost global attractivity for the tracking error variables in the existence of uncertainties. These are verified by rigorous mathematical analysis.

VII. Future Work

The future work in this research work includes comparing this controller response with baseline PID and Model predictive control (MPC) for same aggressive trajectory. Implementing $\mathfrak{L}1$-Adaptive Controller for trajectory optimization experiments on aggressive flights will be one of the agendas. We are looking forward to fill the gap of providing guarantees on robustness by integrating H_2 and $H\infty$ controller with L1-AC to guarantee the robustness of system.

VIII. Acknowledgement

I would like to thank Professor Maxim Likhachev and Ramkumar Natarajan for their guidance throughout this project. I would also like to thank Rachel Burcin, Professor John Dolan, the Search-Based Planning Laboratory, and the entire Robotics Institute community at Carnegie Mellon University, Pittsburgh, PA.

References

[1] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor uav using neural networks," *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 50–66, 2009.

[2] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a quadrotor robot," in *Proceedings Australasian Conference on Robotics and Automation 2006*. Australian Robotics and Automation Association Inc., 2006.

[3] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, p. 7410.

[4] S. Bouabdallah and R. Siegwart, "Towards intelligent miniature flying robots," in *Field and service robotics*. Springer, 2006, pp. 429–440.

[5] A. Sharma and A. Barve, "Controlling of quad-rotor uav using pid controller and fuzzy logic controller," *Int. J. Electr. Electron. Comput. Eng*, vol. 1, no. 2, pp. 38–41, 2012.

[6] C. Liu, S. J. Tang, S. Y. Yang, and J. Guo, "Fuzzy sliding-mode control for quad-rotor trajectory tracking," in *Applied Mechanics and Materials*, vol. 278. Trans Tech Publ, 2013, pp. 1593–1600.

[7] A. Wahyudie, T. B. Susilo, and H. Noura, "Robust pid controller for quad-rotors," *Journal of Unmanned System Technology*, vol. 1, no. 1, pp. 14–19, 2013.

[8] P. Castillo, P. Albertos, P. Garcia, and R. Lozano, "Simple real-time attitude stabilization of a quad-rotor aircraft with bounded signals," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1533–1538.

[9] S. Jafari, P. Ioannou, and L. E. Rudd, "What is l1 adaptive control," in *AIAA guidance, navigation, and control (GNC) Conference*, 2013, p. 4513.

[10] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive control of quadrotor uavs: A design trade study with flight evaluations," *IEEE Transactions on control systems technology*, vol. 21, no. 4, pp. 1400–1406, 2012.

[11] J. M. Selfridge and G. Tao, "A multivariable adaptive controller for a quadrotor with guaranteed matching conditions," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, no. 1, pp. 24–33, 2014.

[12] G. Antonelli, F. Arrichiello, S. Chiaverini, and P. R. Giordano, "Adaptive trajectory tracking for quadrotor mavs in presence of parameter uncertainties and external disturbances," in *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2013, pp. 1337–1342.

[13] H. Zhen, X. Qi, and H. Dong, "An adaptive block backstepping controller for attitude stabilization of a quadrotor helicopter," *Trans. Syst. Control*, vol. 8, no. 2, pp. 46–55, 2013.

[14] E. Xargay, N. Hovakimyan, V. Dobrokhodov, R. Statnikov, I. Kaminer, C. Cao, and I. Gregory, "L1 adaptive flight control system: systematic design and verification and validation of control metrics," in *AIAA guidance, navigation, and control conference*, 2010, p. 7773.

[15] N. HOVAKIMYAN, C. CAO, E. KHARISOV, E. XARGAY, and I. M. GREGORY, "54 l1 adaptive control for safety-critical systems," *IEEE Control Systems Magazine*, 2011.

[16] T. Yucelen and W. M. Haddad, "A robust adaptive control architecture for disturbance rejection and uncertainty suppression with l transient and steady-state performance guarantees," *International Journal of Adaptive Control and Signal Processing*, vol. 26, no. 11, pp. 1024–1055, 2012.

[17] B. S. Kim and A. J. Calise, "Nonlinear flight control using neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 26–33, 1997.

Developing Ethical Agents via Context-Sensitive Modular Inverse Reinforcement Learning

Boshi Wang[1], Yue Guo[2], Dana Hughes[3] and Katia Sycara[3]

***Abstract*— Autonomous robots will soon enter our everyday lives and interact with people, so it is important for these agents to be able to reason about various social norms, moral values, legal rules, and social conventions to gain human acceptability. One approach for this problem in the reinforcement learning framework is through inverse reinforcement learning (IRL), which avoids the difficult task of hand-tuning the reward function for ethical/non-ethical behaviors by learning from ethical demonstrations. IRL-based approaches suffer from the fact that when there are a lot of changing contexts in the environment, the computation would be extremely expensive. Inspired by recent work on Modularized Normative MDPs (MNMDP) and early work on context-sensitive RL in cognitive science literature, we propose a new IRL framework which we call Context-Sensitive Modular IRL (CMIRL). In this model, we add a norm-detection layer on top of states and contexts, resembling human's decision process in a context-aware environment. We assume that the expert constantly determines the importance factor of every norm given the current state and context, where each norm is associated with a reward function over the domain state and the agent chooses actions with high weighted Q-values. The trained model recovers a set of reward parameters for each norm in the environment, and also a module that can estimate relevant importance for each norm under different states and contexts. Our model is capable of handling problems with large context space and the computational complexity is linear in the number of norms in the environment. Simulated experiments validate the effectiveness of our method.**

I. Introduction

Autonomous robots will soon enter our everyday lives and interact with people as co-workers, so it is important for these agents to be able to reason about various ethical norms, moral values, legal rules, and social conventions to gain human acceptability. It has been justified by [1] that the framework of reinforcement learning has the capability of achieving an idealized ethical artificial agent. The idea is that we can incorporate ethical norms into the decision making process of the agent by designing reward/penalty functions corresponding to different types of norm activation in the environment. In this way, reinforcement learning methods that aim to maximize the cumulative reward should enable the agent to behave in an ethical way.

[1]Boshi Wang is with School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China wangbsh@shanghaitech.edu.cn

[2]Yue Guo is with the Computer Science Department, Carnegie Mellon University, Pittsburgh 15213, USA yueguo@andrew.cmu.edu

[3]Dana Hughes, Katia Sycara are with the Robotics Institute, Carnegie Mellon University, Pittsburgh 15213, USA danahugh@andrew.cmu.edu, katia@cs.cmu.edu

A. Challenge of Reinforcement Learning-based Approaches

The reinforcement learning framework for developing ethical agents has two severe challenges. First, it's very hard to manually design appropriate reward functions for ethical behaviors to begin with. As stated in [2], [3], for the agent to behave ethically, we have to enumerate all plausible ethical/non-ethical scenarios or rules and design appropriate and meaningful rewards to each of them. However, unlike traditional RL problems such as chess/Go or Atari games where the reward functions are either given (such as in Atari games) or easy to design (in chess/Go we can give positive reward to winning and vice versa), there's no natural way of designing reward functions for ethical/non-ethical behaviors, let alone designing rewards that can appropriately describe complex scenarios where the agent has to deal with multiple norms at the same time. The second challenge is that, even if we have access to fully-designed reward functions for each scenario, computing the optimal policy for the agent could be extremely costly under a lot of changing contexts. For example, the computational model used in [4] adds all relevant context information into the domain state, and computes the optimal policy via solving the MDP with the augmented states and norm-modified rewards. The problem with this approach is obvious: when there are a lot of environmental contexts, the augmented state space would be too large and the policy computation becomes unfeasible. This is rendered to a certain extent by [5], where they consider the ethical decision making to be a modular process: at each time step the agent could only take the norms that may activate at that specific scenario into consideration, so that although a number of different MDPs at different states and contexts has to be solved, each of the problem would have a smaller size than the problem with fully-augmented state, under the assumption that the number of interacting norms at each scenario is small relative to the full set of possible norms in the environment.

B. Inverse RL for Ethical Learning

The first challenge mentioned above motivates the inverse reinforcement learning (IRL)-based approaches [2], [6], [7], where we can collect enough ethical behavior data, and then recover the reward function which can be used for policy computation. This bottom-up approach avoids hand-crafting the reward functions and is also admired for the ability to generalize to unseen states, and in fact, it's an arguable more natural approach for developing ethical agents: if we look back on how we human ourselves learn to behave ethically, we'll find that to a larger extent, we learn to be ethical by

mimicking what other people do, in other words, learning from demonstrations. However, there are also several existing concerns about IRL-based approaches.

C. Paper Contribution and Outline

The contribution of this paper is two-fold. First, we argue that the existing concerns about IRL-based approaches for developing ethical agents are either not well-justified to be big concerns or able to deal with inside the IRL framework, and instead the biggest challenge of IRL approaches is a computational one, which is the result of the second challenge mentioned above (in brief, if even the forward computation is costly, of course so would the inverse process be). Second, which is our main technical contribution, we propose a new IRL framework which we call Context-Sensitive Modular IRL (CMIRL) motivated from work on Modularized Normative MDP [5], and early work on context-sensitive RL in the cognitive science literature [8]. CMIRL treats states and contexts separately, and assume that the expert constantly determines the importance factor of every possible norm in the environment, where each norm is associated with a different reward function of the domain state and the agent chooses the action with high weighted Q-values. The trained model recovers a set of reward parameters for each norm in the environment, and also a module that can estimate relevant importance for each norm under different states and contexts.

The outline of the paper is as following. Section II discusses about related work on ethical decision making and learning. In section III we introduce some basic background and the general formulation of our problem. In section IV we discuss about concerns and challenges of IRL-based approaches, and section V introduces our modelling of the ethical expert and the corresponding inverse learning method which tackles these challenges. Section VI describes our simulation results and section VII concludes our work and discuss future work directions.

II. Related Work

There are two main branches toward ethical decision making and learning: rule-based approaches and learning-based approaches.

A. Rule-Based Approaches

In rule-based approaches, ethical norms are usually represented by symbolic/logical expressions along with a set of inference rules. For example, In [9] the authors propose using pre-specified logical expression to reason about appropriate ways to accomplish the assigned tasks without violating normative principles. Methods based on Horty logic [10] such as [11], [12] allows reasoning about multiple agents and their actions and compose ethical semantics.

However, rule-based approaches have two severe limitations. First is that uncertainty is not allowed in decision making and norm inference, the second is all the rules must be set in advance and active-learning of these rules from the environment is not possible inside the framework.

B. Learning-Based Approaches

More recent materials have been focusing on methods that allow learning of ethical norms. Work on value alignment [13], [14] tries to align the goals of the agent with that of human's. The reinforcement learning framework is first proposed and justified in [1], where they consider the problem of ethical learning as finding an utility (reward) function that is over the hidden state in a Partially Observable Markov Decision Process (POMDP). The authors of [15] combine the strength of reinforcement learning and logical representations, and propose a hybrid approach where agent maximizes the reward function only over states that satisfy the norms. Although these approaches allow ethical norms to be represented in a much more flexible way, the utility function that induces these ethical behaviors have to be manually designed which could be very challenging in many scenarios. Inverse RL (IRL) based methods that avoid hand-crafting rewards by learning from expert demonstrations are explored in [2], [6], [7], however none of the work focuses on a computational perspective where big challenge lies when IRL-based approaches are put into real applications, which is the major focus of this paper.

III. Background and Problem Formulation

A. Markov Decision Process

Definition 1 (Markov Decision Process)**.** *A Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is the reward function, $\mathcal{T}(s, a, s') = p(s'|s, a)$ is the probability that the agent takes action a at state s and transits to s' in the next time step, and $\gamma \in [0, 1]$ is the discount factor.*

A *policy*, $\pi(s, a) = p(a|s)$, specifies the probability of performing action a at state s. Under a given policy π, the value of a state $V^\pi(s)$ is defined by the expected total discounted reward obtained starting from state s:

$$V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)|s_0 = s, \pi] \tag{1}$$

and similarly the Q-function $Q^\pi(s, a)$ is defined by the expected total discounted reward obtained starting from state s and performing action a:

$$Q^\pi(s, a) = \mathbb{E}[\mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^\pi(s')] \tag{2}$$

The advantage function of a state-action pair under policy π is defined to be:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{3}$$

which captures how much better an action a is than the "average action" under π at state s.

The optimal policy π^* is the policy that maximizes the discounted reward received for any state. We denote the corresponding value function, Q-function and advantage function under policy π^* as V^*, Q^*, and A^* respectively.

B. Inverse Reinforcement Learning

In many problems, the reward function is very difficult to manually specify, and instead we have access to expert demonstrations. This motivates Inverse Reinforcement Learning (IRL). First proposed in [16], IRL aims to find the expert's reward function given the it's trajectories where each trajectory consists of a sequence of state-action pairs. It goes one step further from imitation learning (IL) where the objective is usually to learn a policy. The advantage of IRL over IL is that, the reward function, rather than the policy, is the most robust and transferable definition of the task, and usually it takes a relatively compact structure [17] which has very high interpretability and makes subsequent fine-tuning easy ro perform. In contrast, the typical result of IL is a deep policy network which is extremely hard to interpret and tune. This is also the reason why imitation learning approaches are generally not considered in ethics learning.

Through about 20 years of research, a big family of IRL algorithms are developed and studied which are summarized in [18]. This paper has no intention of improving basic IRL methods, and instead we use them as part of the building blocks to construct learning methods in more complicated models. Our work is primarily based on maximum likelihood IRL (MLIRL) [19], [20], but it should not be restricted to other types of IRL algorithms which is not the focus of this paper.

C. Problem Formulation

The scenario we are working with extends MDP with additional context information. While modelling the agent's decision process in a context/norm-aware environment is in fact a part of the problem solving routine, defining the general formulation of the problem is easy and clear to begin with. We use $\mathcal{C}$ to denote the context space and c_t to denote the specific context the agent observes at time step t. The problem we study is:

We are given a set $D = \{\tau_1, ..., \tau_N\}$ of N expert demonstrations where τ_i is the ith trajectory. A trajectory τ consists of T_τ state-context-action pairs $\{(s_1, c_1, a_1), ..., (s_{T_\tau}, c_{T_\tau}, a_{T_\tau})\}$. We are not aware of the model of the expert, except that the expert's action is independent of any other information given the current state and context. Our goal is to learn a reward function under which the optimal policy $\pi^*(a|s, c)$ induces behaviors that are similar to the expert's.

Note that the above definition we propose is a very general formulation, and solving this problem in specific domains involves appropriate modelling of the expert's decision process in the corresponding scenarios. This paper targets at modelling an ethical agent in a norm-aware setting, for which the details will be covered in later sections.

IV. IRL for Ethical Agents: What's the Real Challenge?

In this section, we argue that the existing concerns for adopting IRL-based approaches are inadequate, and make the point that the biggest problem of IRL approaches is the computational cost under large context space, which is the problem we try to tackle in section V.

- In [3], the author states that collecting a large amount of human data toward maximizing the reward could be costly. We believe that this is not a well-justified problem: collecting enough amount of data could indeed be costly, but this doesn't suggest IRL's in-adequateness. In analogy, the progress of deep learning-based methods which arises dramatically these years also rely hugely on large amount of training data, for which the collecting processes were also costly. If training a successful digit classifier would require a great amount of data, it is natural that we should collect a lot of data for developing an ethical agent.
- Both [3] and [15] mentioned that the data may be biased and if an IRL agent learns from unethical behavior, it will learn to behave unethically. We don't consider this to be an appropriate concern especially in the context of developing ethical agents. The reason is simple: unlike in many traditional RL problems where the optimal behavior is given by the environment which could be out of human's controls, whether a behavior is ethical or not is entirely up to ourselves. Therefore we are completely able to tell what behaviors are ethical and what are not and provide fully ethical behavior data. If we ourselves don't even make the efforts to show the agent what we consider to be ethical, we can't expect the agent to learn to behave in our desired way. This argument also justifies that IRL is a desirable approach toward developing ethical agents, since in this case we human are able to provide optimal demonstrations unlike many other problems.
- The author of [21] mentioned that IRL is not able to infer temporally complex norms since it's based on the Markovian assumption, namely the activation of norms only depends on the current state and contexts. The author also stated that this could be handled by using models with internal memory such as recurrent neural networks to remove the Markov assumption, but the lack of interpretability of this approach may make it unsuitable for ethics. While the interpretability of deep networks is still an active area of research, these networks are computationally scalable in a practical perspective and have good performance in many fields. In contrast, methods using symbolic inference such as the one proposed by the same author are not computationally scalable and therefore couldn't work in large scale applications. As we will show, norms that are non-Markovian can be dealt with by a very simple modification to our proposed model.

The above arguments eliminate the existing concerns of IRL framework for developing an ethical agent. However, we'd like to make the point that the biggest challenge of IRL is its computational complexity under the environment with a lot of different and maybe irrelevant contexts.

Specifically, given the problem defined in section III, a

direct model of the expert would be the one given in [4], which assumes that the expert treats the state and context as a whole, called augmented state, and follows the optimal policy induced by some reward function in this "augmented" MDP. This model of the expert reduces our problem of study to a simple basic IRL problem and can be solved via any existing IRL method. When the augmented state space is small, this method works perfectly; however in scenarios where the agent have to deal with a lot of changing context information, this approach is clearly intractable. Therefore we have to make a more compact model about how the context involves in the expert's decision process. We make a first attempt in this problem which is described in section V.

V. Technical Approach

A. Modelling the Expert: Context-Sensitive Modular MDP

We first discuss about how essentially can we achieve reduction of computational complexity. To summarize this at a high level, the reduction could be made by sacrificing the prediction of future contexts, which also is the essential idea (although not explicitly stated by the author) of the modularity in [5]. Recall the naive approach mentioned earlier. Running IRL on the "augmented" MDP would implicitly involve estimating the transition probabilities with the augmented states, which includes the transition of contexts. However, sacrificing this part shouldn't harm much the performance. If we think back about the decision process of ourselves in daily lives, for example, when we are walking towards some destination and suddenly we notice that a person (not necessarily on our path) is in trouble and needs help, so we abandon the original route and instead go and help that person. Do we constantly calculate the probability that there will be a person who needs help in the next time step, or do we simply react on what we observe? It's obvious that the latter one fits better to our own experience. Another consideration is, in certain scenarios the contexts are simply not predictable so estimating the context transitions are not helpful, for example when a car suddenly runs into someone's sight, the person stops immediately to avoid crashing. These all suggests that an intuitive model is to treat the contexts as something that we just observe and not predict, and that the expert's entire current policy should be, in some sense, **conditioned** on the current state and context so it becomes a modular decision process where the policy of the agent changes when the states/contexts change.

Now we add the notion of ethical norms into consideration. It is often that the expert have to deal with multiple norms at the same time. For example, when we are on the way helping someone, we also have to be careful not to crash into people, so two completely different norms ("help" and "avoid") must be simultaneously considered. Therefore, the priority of norms must be taken into account, and we also need to learn this priority from the expert's behaviors.

Two existing work are most closely related to the above intuitions. The first one is Contextual MDP (CMDP) [22]. In CMDP there is a mapping $\mathcal{M}$ which maps a context c to an MDP with its own reward functions and transitions. In our problem this mapping should also take current domain state into consideration. The second one is Modular MDP [23]–[26] where the reward function is usually expressed as a convex combination (with fixed weights) of a set of reward modules. In our problem, each module in Modular RL could be regarded as describing a certain type of norm in the environment, but the contexts are not considered and the weights are fixed which is a severe limitation.

While CMDP is a too general model and Modular MDP is too limited for our problem, we perform an elegant combination of these two models with appropriate modifications. The new model of the expert, which we call Context-Sensitive Modular MDP, generalizes Modular MDP with dynamic weights determined by a mapping $\mathcal{M}$ from current state and context:

Definition 2 (Context-Sensitive Modular MDP)**.** *A Context-Sensitive Markov Decision Process is a tuple $(\mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{M}, \{\mathcal{R}_k\}_{k=1,..,K}, \gamma)$, where besides the components in MDP (now with K reward functions instead of 1), $\mathcal{C}$ is the context space and $\mathcal{M}(s,c)$ maps a (state, context) tuple to a vector of importance factors $\mathcal{M}(s,c) \in \mathbb{R}^K$ where $\mathcal{M}(s,c) \succeq 0$ and $\mathbf{1}^T\mathcal{M}(s,c) = 1$. Under state s and context c, the agent acts in an MDP with reward function defined by $\sum_{k=1}^{K} \mathcal{M}(s,c)_k \mathcal{R}_k$, and after each action new state and context are observed.*

Intuitively, in our problem, each reward function corresponds to a norm in the environment, and the mapping $\mathcal{M}$ decides each norm's relevant importance given the state and context. Note that for norms that are non-Markovian [21], we can simply alter the mapping $\mathcal{M}$ to depend on historical information such as all past states and contexts, which can be represented by, for example, recurrent networks.

We assume that the expert demonstrations described in section III are generated from this process, and our goal is to learn, from these demonstrations, the reward functions and the mapping $\mathcal{M}$.

B. Inverse Learning via Maximum Likelihood Estimate

We use parametrized function approximators to represent each reward function and the mapping $\mathcal{M}$. We will call the reward parameters $\theta_1, ..., \theta_K$ (θ_k corresponds to reward function $\mathcal{R}_{\theta_k}$), and the mapping $\mathcal{M}_\alpha(s,c)$ with parameters α. θ is shorthand for $\{\theta_1, ..., \theta_K\}$.

Following the tradition in IRL methods we use Boltzmann distribution for the expert's policy [27]. Illustration of the expert's whole decision process is depicted in Fig.1.

We use the principle of maximum likelihood to estimate the parameters in the model. Assuming the trajectories are generated independently, we can write the total log-likelihood of D:

$$\log L(D) = \sum_{\tau \in D} \log p(\tau) \tag{4}$$

where for $\tau = \{(s_1, c_1, a_1), ..., (s_{T_\tau}, c_{T_\tau}, a_{T_\tau})\}$:

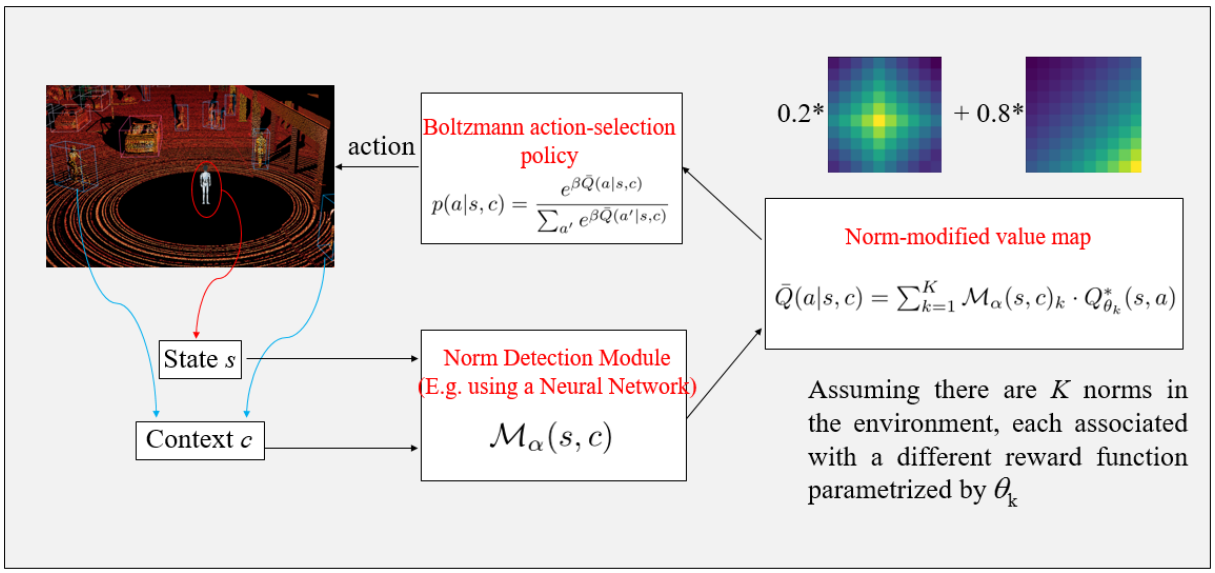


Fig. 1: Illustration of Context-Sensitive Modular Markov Decision Process. At first, the agent perceives the current state s and environmental context c. Then, the agent estimates each norm's importance factor given s, c using the norm-detection module, which results in a vector of importance factors $\mathcal{M}(s,c)$. Then the agent computes the weight Q-value map and chooses action with Boltzmann policy, and transits to a new environment where the process goes on repeatedly.

$$
\begin{aligned}
p(\tau) =& p(s_1)p(c_1)p(a_1|s_1,c_1) \\
& * \prod_{t=2}^{T_\tau} p(s_t|s_{t-1},a_{t-1})p(c_t)p(a_t|s_t,c_t)
\end{aligned} \tag{5}
$$

Let $\bar{Q}(s,a|c)$ be the weighted optimal Q-value of (s,a) when the agent is at state s with context c:

$$
\bar{Q}(s,a|c) = \sum_{k=1}^{K} \mathcal{M}_\alpha(s,c)_k \cdot Q^*_{\theta_k}(s,a) \tag{6}
$$

the probability that the agent will perform action a is given by:

$$
p(a|s,c) = \frac{e^{\beta \bar{Q}(s,a|c)}}{\sum_{a'} e^{\beta \bar{Q}(s,a'|c)}} \tag{7}
$$

where β controls the extent to which the agent prefers actions with high Q-values. When $\beta \to \infty$, this policy selects the action with maximum Q-value, and when $\beta = 0$, this policy chooses actions at uniform. The summation in the denominator is over valid actions (a' in the above formula) at state s, which is dropped for simplicity.

Now we can write down the log-likelihood of demonstrations D:

$$
\begin{aligned}
& \log L(D;\alpha,\theta) \\
=& \sum_{\tau \in D} \log p(s_1) && (a) \\
& + \sum_{\tau \in D} \sum_{t=1}^{T_\tau} \log p(c_t) && (b) \\
& + \sum_{\tau \in D} \sum_{t=2}^{T_\tau} \log p(s_t|s_{t-1},a_{t-1}) && (c) \\
& + \sum_{\tau \in D} \sum_{t=1}^{T_\tau} \log p_{\alpha,\theta}(a_t|s_t,c_t) && (d)
\end{aligned} \tag{8}
$$

Terms $(a), (b), (c)$ each concerns a unique parameter type and therefore these parameters can be estimated in separate very easily. Therefore we assume we have access to the initial state distribution $p(s_1)$, the context probabilities $p(c_t)$, and the transition probabilities $p(s_t|s_{t-1},a_{t-1})$. By our former argument, although we can estimate $p(c_t)$, it shouldn't be used which is indeed the case in our approach.

Concerning only term (d), we need to calculate the gradient of $\log p_{\alpha,\theta}(a_t|s_t,c_t)$ with respect to each θ_k and α so that we can perform gradient ascent. For the reward parameters θ_k, expanding out the terms in Equation (6) and (7) and with some simplifications we have:

$$
\begin{aligned}
\nabla_{\theta_k} \log p_{\alpha,\theta}(a_t|s_t,c_t) =& \beta \mathcal{M}_\alpha(s_t,c_t)_k (\nabla_{\theta_k} Q^*_{\theta_k}(s_t,a_t) \\
& - \sum_{a'_t} p_{\alpha,\theta}(a'_t|s_t,c_t) \nabla_{\theta_k} Q^*_{\theta_k}(s_t,a'_t))
\end{aligned} \tag{9}
$$

From this we can see that we need to compute $\nabla_{\theta_k} Q^*_{\theta_k}(s,a)$ for each k, which can be estimated by solv-

ing a set of fixed-point equations similar to the Bellman-optimality equations [28]:

$$\nabla_{\theta_k} Q^*_{\theta_k}(s,a) = \nabla_{\theta_k} \mathcal{R}_{\theta_k}(s) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} \sum_{a'} \pi^*_{\theta_k}(s',a') \nabla_{\theta_k} Q^*_{\theta_k}(s',a') \tag{10}$$

For α we have:

$$\nabla_\alpha \log p_{\alpha,\theta}(a_t|s_t,c_t) = \beta \sum_{z_k} (Q^*_{\theta_k}(s_t,a_t) - \sum_{a'_t} p_{\alpha,\theta}(a'_t|s_t,c_t) Q^*_{\theta_k}(s_t,a'_t)) \nabla_\alpha \mathcal{M}_\alpha(s_t,c_t)_k \tag{11}$$

where computation of $\nabla_\alpha \mathcal{M}_\alpha(s_t,c_t)_k$ depends on specific models being used, for example when $\mathcal{M}_\alpha$ is represented by a neural network, this gradient can be computed by standard back propagation.

An insightful observation from equation (11) is that, the term $(Q^*_{\theta_k}(s_t,a_t) - \sum_{a'_t} p_{\alpha,\theta}(a'_t|s_t,c_t) Q^*_{\theta_k}(s_t,a'_t))$ is exactly the conditional advantage under context c_t corresponding to the kth reward function: $A^{\pi_{\theta_k}}(s_t,a_t|c_t)$. Therefore we can rewrite this gradient by:

$$\nabla_\alpha \log p_{\alpha,\theta}(a_t|s_t,c_t) = \beta \nabla_\alpha \sum_{k=1}^{K} A^{\pi_{\theta_k}}(s_t,a_t|c_t) \mathcal{M}_\alpha(s_t,c_t)_k \tag{12}$$

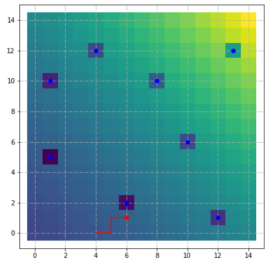
which is a very intuitive formulation of what gradient ascent on α does under our model: at each step, α is optimized so as to **maximize the sum of weighted conditional advantage** of all (s,a,c) pairs in the expert trajectories under current parameters.

VI. Simulations and Results

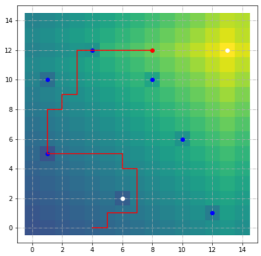
We validate the effectiveness of our method using simulated experiments. Note that we have no intention of performing large-scale simulations, and instead we try to stress the core challenge (large number of changing contexts) and make other components as simple as possible to emphasize the central spirit of our model.

A. Setup

In the experiment, we extend scenario "Grab a Milk" in [3] to a **dynamic** setting which is a lot more challenging and realistic. In a 15 by 15 map, the agent starts from (4, 0) and the domain task is to go to the goal position (14, 14) (the upper right corner) to get milk. The agent can move in 4 directions (except in sides/corners) and we use deterministic transitions for simplicity. There are a number of babies on the map and they may start crying at **random** time steps. The agent has to help (soothe) the babies who are crying by going to the babies' locations. For each time step, the domain state is the agent's location, and the context is each baby's status (1 for crying and 0 for being quiet) and position (fixed). If no babies are crying, the expert would go to the goal position and try to avoid the babies (the reward is +1 at goal, -1 at baby locations and 0 for others); otherwise the expert will always go help the nearest crying baby (the reward is +3 at nearest baby's location, -1 for other babies, +1 at goal and 0 for others). We set $\beta = 7$. Note that in this setting, the size of context space is **exponential** in the number of babies and the context are almost constantly changing, but the number of reward functions is small: the agent is either helping one of the babies or going to the goal, so the number of reward functions K grows linearly with the number of babies put into the map. Fig.2 shows two expert behaviors under different scenarios, with a setting of 8 babies that are located at random positions.



(a) Initially, no baby is crying, so the expert goes toward the goal (upper right corner) while trying to avoid crashing into the babies.



(b) In this case, two babies are crying (represented by white dots). The expert goes toward the nearest baby to help while also trying to avoid crashing into other babies. The two blue babies on the agent's trajectory have been soothed by the expert (instead of crashed into).

Fig. 2: Example of the expert's decisions in different scenarios used for generating demonstration trajectories. States with light color have relatively high rewards and states with darker color have lower rewards. The red line shows the trajectory of the agent and red dot is the current position of the agent.

We set each reward function to take the form $\mathcal{R}_{\theta_k}(s) = \theta_k^T \phi(s)$, where $\phi(s)$ is a binary vector consisting of whether the agent is at each baby's position and goal position, so the dimension of $\phi(s)$ is (number of babies + 1). We simulate using 8 babies on the map (as the one in Fig.2), where the time step that each baby begins crying is chosen uniformly from 0 to 40. 500 expert trajectories are generated. We set $K = 10$ and the architecture of $\mathcal{M}_\alpha$ to be 16-20-10 (number of units in different layers) with L_2 regularity factor of 10^{-5} on the weight matrices, where the input to the network consists of each baby's crying status and its Euclidean distance to the agent. Again, in real applications, these models can be easily changed to be very complicated with large scale.

B. Evaluation Metrics and Results

We evaluate the similarity of the learned behavior and the expert's behavior by calculating the average number of times the agent helps and crashes into babies during one task. For every 10 iterations of training, we sample 500 trajectories

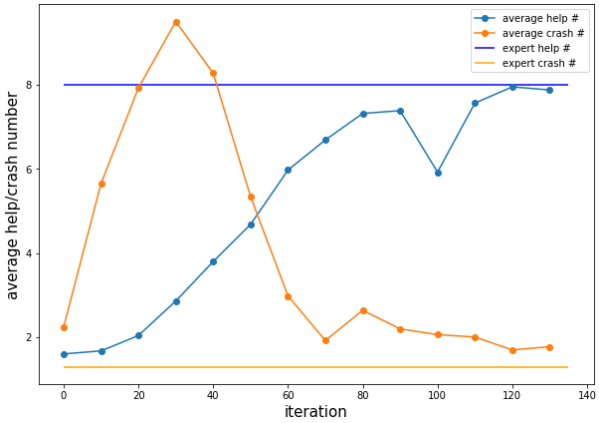


Fig. 3: Average number of times the agent helps/crashes into the babies in the environment during different training iterations, compared with the expert's.

using the parameters and compute these numbers, which are plotted in Fig.3 together with the expert's average number of helps and crashes.

It can be seen from Fig.3 that the average number of helping steadily increases during training, and although at first the agent doesn't pay attention to avoid crashing into babies (which may be because the agent tries very hard to help babies and ignores other factors), this is taken into account after a number of iterations. The agent's behavior is ultimately aligned with the expert's, showing our method's capability of learning ethical behaviors under big amount of changing contexts.

VII. Conclusion and Future Work

We propose Context-Sensitive Modular Inverse Reinforcement Learning, a framework for developing ethical agents through learning from expert demonstration in an environment with large number of changing contexts. In the future work, we intend to implement our methods in real and large scale applications, and perform sample complexity analysis of our model.

Acknowledgement

Boshi Wang thanks Katia Sycara, Dana Hughes, Yue Guo for guidance, and ShanghaiTech University for funding the work. He also gives special thanks to Dr. John Dolan, Ms. Rachel Burcin and the RISS team for support.

References

[1] D. Abel, J. MacGlashan, and M. L. Littman, "Reinforcement learning as a framework for ethical decision making," in *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[2] C. E. Sezener, "Inferring human values for safe agi design," in *International Conference on Artificial General Intelligence*. Springer, 2015, pp. 152–155.

[3] Y.-H. Wu and S.-D. Lin, "A low-cost ethics shaping approach for designing reinforcement learning agents," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] M. S. Fagundes, S. Ossowski, J. Cerquides, and P. Noriega, "Design and evaluation of norm-aware agents based on normative markov decision processes," *International Journal of Approximate Reasoning*, vol. 78, pp. 33–61, 2016.

[5] V. Krishnamoorthy, W. Luo, M. Lewis, and K. Sycara, "A computational framework for integrating task planning and norm aware reasoning for social robots," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018, pp. 282–287.

[6] K. Amin and S. Singh, "Towards resolving unidentifiability in inverse reinforcement learning," *arXiv preprint arXiv:1601.06569*, 2016.

[7] O. Evans, A. Stuhlmüller, and N. Goodman, "Learning the preferences of ignorant, inconsistent agents," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[8] C. Balkenius and S. Winberg, "Cognitive modeling with context sensitive reinforcement learning," *Proceedings of AILS*, vol. 4, pp. 10–19, 2004.

[9] G. M. Briggs and M. Scheutz, "sorry, i cant do that: Developing mechanisms to appropriately reject directives in human-robot interactions," in *2015 AAAI fall symposium series*, 2015.

[10] J. F. Horty, *Agency and deontic logic*. Oxford University Press, 2001.

[11] K. Arkoudas, S. Bringsjord, and P. Bello, "Toward ethical robots via mechanized deontic logic," in *AAAI fall symposium on machine ethics*, 2005, pp. 17–23.

[12] S. Bringsjord, K. Arkoudas, and P. Bello, "Toward a general logicist methodology for engineering ethically correct robots," *IEEE Intelligent Systems*, vol. 21, no. 4, pp. 38–44, 2006.

[13] S. Russell, D. Dewey, and M. Tegmark, "Research priorities for robust and beneficial artificial intelligence," *Ai Magazine*, vol. 36, no. 4, pp. 105–114, 2015.

[14] N. Soares and B. Fallenstein, "Aligning superintelligence with human interests: A technical research agenda," *Machine Intelligence Research Institute (MIRI) technical report*, vol. 8, 2014.

[15] T. Arnold, D. Kasenberg, and M. Scheutz, "Value alignment or misalignment–what will keep systems accountable?" in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[16] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[17] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[18] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *arXiv preprint arXiv:1806.06877*, 2018.

[19] M. C. Vroman, "Maximum likelihood inverse reinforcement learning," Ph.D. dissertation, Rutgers University-Graduate School-New Brunswick, 2014.

[20] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 897–904.

[21] D. Kasenberg, T. Arnold, and M. Scheutz, "Norms, rewards, and the intentional stance: Comparing machine learning approaches to ethical training," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 2018, pp. 184–190.

[22] A. Hallak, D. Di Castro, and S. Mannor, "Contextual markov decision processes," *arXiv preprint arXiv:1502.02259*, 2015.

[23] E. Uchibe, M. Asada, and K. Hosoda, "Behavior coordination for a mobile robot using modular reinforcement learning," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, vol. 3. IEEE, 1996, pp. 1329–1336.

[24] N. Sprague and D. Ballard, "Multiple-goal reinforcement learning with modular sarsa (0)," 2003.

[25] S. J. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 656–663.

[26] C. A. Rothkopf and D. Ballard, "Credit assignment in multiple goal embodied visuomotor behavior," *frontiers in Psychology*, vol. 1, p. 173, 2010.

[27] G. H. John, "When the best move isn't optimal: Q-learning with exploration," in *AAAI*. Citeseer, 1994, p. 1464.

[28] G. Neu and C. Szepesvári, "Apprenticeship learning using inverse reinforcement learning and gradient methods," *arXiv preprint arXiv:1206.5264*, 2012.

Traversability Analysis for Highly Maneuverable Wheeled Robots

Letong Wang[1], Sean Wang[2] and Aaron Johnson[3]

Abstract— **Outdoor robots capable of traversing over rough terrain could play an essential role in tasks such as field rescue, oil prospecting, and exploration of abandoned mines. Many algorithms currently enable robots to autonomously navigate through outdoor terrains, such as city roads and country trails. However, most methodologies for navigation are designed for robots working in structured environments or based on existing maps. In this project, we use a highly maneuverable wheeled robot that can go over unstructured outdoor environments, such as off-road and off-trail environment, where there might be small holes and grooves. For off-trail navigation, terrain traversability analysis is crucial for safety and efficiency. To accomplish this, some methods are proposed that are based on geometry-based, appearance-based, and robot-based features. However, small negative objects (e.g. small holes), that do not affect the robot's traversability, are incorrectly diagnosed as untraversable. We propose the "wheel dropping" model, which considers both geometric features and the robot's capabilities, to smooth the original elevation map. Then, a series of filters are applied to the elevation map to extract geometric features, such as slopes and roughness, to construct the traversability map. This new "wheel-dropping" method allows the robot to traverse over small negative objects that were falsely identified as untraversable by previous traversability analysis methods.**

I. Introduction

Autonomous navigation in off-trail environments has many applications, such as field rescue, oil prospecting and exploration of abandoned mines. Despite recent advances, navigating in unstructured outdoor environments for autonomous mobile robots remains a significant challenge due to the complexity of the environment. To avoid collisions, robots need local maps to tell themselves the location of obstacles and free space. In indoor environments, the grounds on which robots are running are assumed to be planar, and a 2D local map can be constructed by projecting the range scans to the ground. In outdoor environments, 3D analysis of terrains and the capacity of robots are required to determine the probable places through which the robot could traverse. Particular difficulties are negative obstacles (characterized by absence of supporting ground, e.g. holes), hanging obstacles (whose projection onto the ground stands inside the robot's region but far above the robot, e.g. overhangs) and vegetation obstacles (which are flexible so that they can be rolled over, e.g. high grass). Since our robot is meant to operate in a rocky desert environment, the last one is ignored.

[1]L. Wang is with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China wanglt@shanghaitech.edu.cn

[2]S. Wang is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA seanjwang@cmu.edu

[3]A. Johnson is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA amj1@cmu.edu

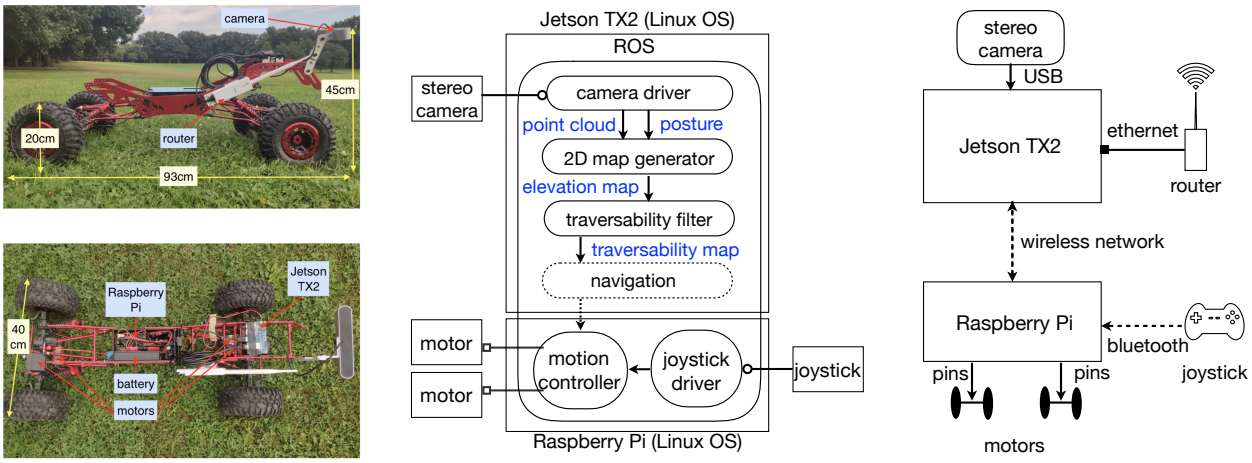
Traversability analysis of the terrain is critical for navigation in unstructured outdoor environments. Traversability is generally defined as an affordance that is jointly determined by the robot's capabilities and its environment's characteristics. The traversability of a particular terrain describes probability for a specified robot to safely maneuver over it. The estimation of traversability is vital for outdoor navigation. If the terrain traversability is underestimated, the robot may have to do unnecessary bypasses or even be trapped, while if the traversability is overestimated, the robot may go somewhere dangerous and damage itself.

In this project, we are given a highly maneuverable four wheel steering robot, shown in figure Fig.1(a), tasked with travelings over unstructured rocky desert terrain to reach target locations. This paper focuses on the real-time construction of the traversability map used for local guidance. A typical way to determine whether an area is traversable is to analyze geometric features, such as slopes and roughness. One common feature in rocky desert terrain are holes and grooves. These features may either have a height significantly lower than its surroundings, or be represented as "empty" due to occlusion of the sensor. If a hole is small compared to the size of the robot wheel, the robot could go over it smoothly. However, because of the "empty" reading or the high slope and roughness around the hole, the hole will likely be classified as untraversable by traditional methods. In this paper, we propose the "wheel dropping" model, which is used to smooth the original elevation map before computing its slopes and roughness. Instead of calculating slopes and roughness based on the actual elevation of the terrain, this model considers the elevation of the center of the wheel as it would sit on the terrain.

We evaluate our methods both on simulated data and real-world data. Compared with methods without "wheel dropping" model, our method diagnoses small negative objects more traversable, which is closer to the reality.

II. Related Work

Lots of traversability analysis methodologies are based on an assumption that the ground is roughly flat, at most has modest dips and rises. The authors in [3] proposed a system that utilized this analysis tool to review the terrain. Firstly, they segment pixels by planes belonging to the same obstacle according to RGB information. Then, they use depth information to determine the plane most likely to be the ground. After that, they set a threshold and considers all pixels with distances to the ground less than the threshold belonging to the ground, which means these points are traversable. Yet, this method does not consider the robot's

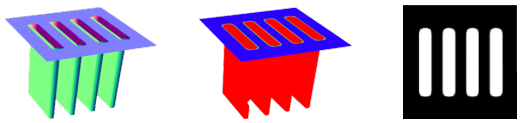


(a) Robot outlook (b) software configuration (c) hardware configuration

Fig. 1. Robot system configuration

capability parameters. The authors in [7] set the threshold according to the maximum height the robot can go over, which depends on the motor torque, wheel diameter, and robot weight. Furthermore, they remove out some outlayers, such as fallen leaves, by their features observed in laser scan data. The authors in [1], besides ground detection and setting two thresholds for objects identify, also detect free sight line in disparity space.

The assumption that the ground is roughly flat does not hold when working with uneven terrains, such as rocky deserts, which has steep hills and grooves. Instead of modeling the ground as a flat plate, some approaches refined the modeling by considering geometric features, such as roughness and slopes. The authors in [2] think a robot's traversability mainly depends on three factors: the maximum step height that the robot can climb, the maximum the slope the robot can climb or descend and the height of the platform. Therefore, they try to extract height difference features and slopes. They firstly compute the norm for every 3D point using the depth image, and then store the data via a two-layer hash-table like structure. Then they take the platform height into account, by diagnosing any points whose difference in z coordinate to the previous obstacle is larger than the platform's height as traversable. They detect untraversable steps by analyzing the height differences between neighbours and they extract slopes by the norm computed before. This approach labels points as traversable, untraversable and uncertain, which works well for stairs, little rocky surface, and niches. In this approach, for v-shaped valley whose slope is traversable expect a corner-shaped bottom, which should not be untraversable. The authors in [6] also consider slopes, edges and roughness as important geometric factors for traversability. Instead of storing data in a particular data structure for efficient access to neighbours, they convert raw 3D point cloud data to elevation maps and store them in a universal grid map. They compute the slopes and roughness according to the elevation map, and use a weighted sum of these two features as the traversability. According to the paper, this approach works well in the unstructured indoor environment. However, we find that, as shown in Fig.2, as long as the grooves are deep enough, the algorithm will diagnose them as untraversable no matter how small the gaps are, because they have significant slopes and roughness. Although geometric features are vital for traversability, it is determined jointly by geometry features and the robot's capabilities. Some gaps are small compared to the size of the robot's wheel, which should be diagnosed as traversable, as the robot could go over them easily.



(a) color norm (b) traversability (c) grid map

Fig. 2. some grooves are diagnosed as untraversable, although their widths are small compared to the robot's wheel and the robot can easily go over them.

Negative objects are crucial for outdoor navigation, which may lead to hazards. The authors in [8] consider gaps in outdoor terrain traversability. Their methodology has two steps: the first is to detect gaps, the second is to analyze the traversability of gaps individually. They detect gaps based on the assumption that the gap contour satisfies a plane up to a constant error in terms of its distance to the ground. Firstly, they find out the ground equation in front of the robot, and then, for every point in the 3D point cloud, they determine whether a point belongs to the ground, a positive object, or a negative object. Then they uses a contour detecting algorithm to detect gap contours, and assembles the gap point clouds G_j. For the traversability part, they apply Principle Component Analysis (PCA) to help determining the narrowest width of the gap and evaluates traversability for every gap based on the robot's capability. Each gap G_j is represented by four parameters: the center of the gap, whether it is traversable, the start and end points for the traverse path if it is traversable. However, the assumption

this method based may not be satisfied in pretty uneven terrain, and the method separates the traversability analysis for gaps from the whole terrain, which brings in additional computation.

III. Method

Our approach is based on [6], but dealing with the negative object issue. Instead of detecting holes and analyzing them individually as in [8], we apply a wheel-based filter to "fill" small holes. As the robot can move in any direction, we model the wheel to a ball, which is a simplified model. The algorithm works like dropping balls on the ground, and generating a new surface consisting of the center of balls.

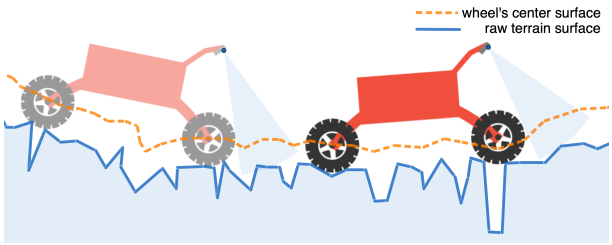


Fig. 3. The wheel's center surface is much smoother than the raw terrain surface, where small holes are "filled".

A. "Wheel Dropping" Model

As shown in Fig.3, we observe that the surface consisting of the centers of the robot's wheels as they would sit on the ground, is much smoother than the raw terrain surface, where small holes disappear. Inspired by that, we build a new surface based on the actual position of the robot's wheel on the ground.

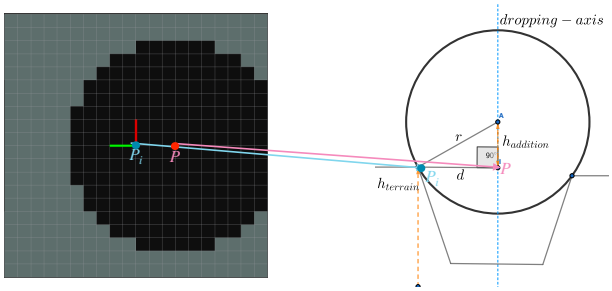


Fig. 4. "wheel dropping" model

The actual location of the robot's wheel as it would sit at a certain point on the ground, is the same as that of a wheel dropped along the z-axis at that point. And because the wheel can move in any direction, we simplify it to a ball with the same radius r.

The algorithm diagram is shown in Fig.4. For a certain position P in the grid map, consider dropping a ball along the z-axis at P, and the ball touching the ground at its neighbor P_i. The height of the ball's center is computed by $terrain(P_i) + \sqrt{r^2 - distance(P, P_i)^2}$. When dropping a ball, we image that the ball will stop the first time it touches the ground, so that the maximum height computed among all the neighbors in radius r is the actual height the dropped ball.

$$new(P) = \max_{P_i \in \mathbf{C}(P,r)} terrain(P_i) + \sqrt{r^2 - dist(P, Pi)^2}$$

where $terrain(P)$ represents the height of P in the raw elevation map, $new(P)$ represents the height of P in the "wheel dropping" surface. $\mathbf{C}(P, r)$ represents a circle-shaped sliding window in the raw elevation map, whose center is P and the radius is r.

Use this window sliding all over the original terrain map, we get a smoother surface, where small holes are "filled".

B. Traversability Analysis

Given an elevation map smoothed by the "wheel dropping" model, a series of filters are applied to the map to extract geometric features and calculate the traversability.

- mean in radius filter:
  Compute the average height within a circle-shaped window as the value of the center point, which smooths the map and will be used to extract the roughness feature.
- surface norm:
  The surface norm is generated by computing the eigenvector within a sliding window corresponding to smallest eigenvalue, which is used for extracting the slope feature.
- roughness:
  roughness is computed by

  $$roughness = \|\text{smoothed} - \text{non-smoothed}\|$$

  where "non-smoothed" represents the layer before applying "mean-in-radius" filter, and "smoothed" represents the layer after applying that filter. The greater the roughness is, the less traversable the terrain is.
- slope:
  The slope is computed by

  $$slope = arccos(norm_z)$$

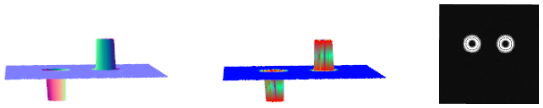
  where the slope here is represented by the angle between the surface norm and the z-axis, which is computed by the arccos value of the norm vector in z coordinate. Slopes range from 0 to $\frac{\pi}{2}$, and the smaller the slope value is, the easier it is for the robot to pass through.
- traversability:
  Based on the features generated above, the traversability map is computed by

  $$1 - a * slope - b * roughness$$

  where a and b are positive parameters for slope and roughness, which depends on the robot's capabilities, including the maximum step height that the robot can climb and the maximum the slope the robot can climb or descend.

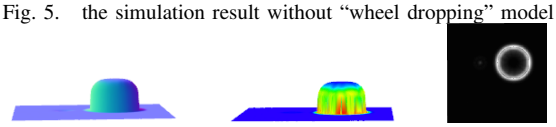
IV. Simulation Experiments and Results

We use gray-scale images to generate simulation elevation maps, by linearly mapping gray value to the elevation value. We apply the traversability analysis on different terrains, and compare the result with and without the "wheel dropping" model.



(a) color norm (b) traversability (c) grid map

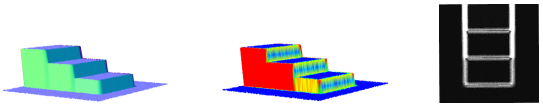
Fig. 5. the simulation result without "wheel dropping" model



(a) color norm (b) traversability (c) grid map

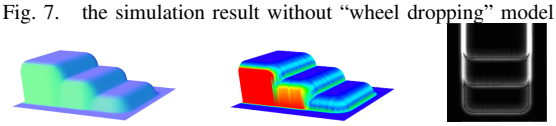
Fig. 6. the simulation result with "wheel dropping model"

1) a hole and a pole: In this simulation terrain, there is a hole and a pole with the same radius(10cm). The radius of the robot's wheel is greater(15cm), and the robot could pass the hole easily, but it should avoid the pole. Without the "wheel dropping" model, as shown in Fig.5(c), the traversability for the pole and the hole are the same, which are both untraversable. Fortunately, as shown in Fig.5(a), the hole was "filled" after filtered by the "wheel dropping" model. In Fig.6(c), the traversability for the pole and the hole are different, and the hole is diagnosed as traversable.



(a) color norm (b) traversability (c) grid map

Fig. 7. the simulation result without "wheel dropping" model



(a) color norm (b) traversability (c) grid map

Fig. 8. the simulation result with "wheel dropping" model

2) stairs: In this simulation terrain, there are stairs, whose height is the same as the radius of the wheel, and our robot could go over them. Nevertheless, without the "wheel dropping" model, as shown in Fig.7(c), the edges of the stairs are diagnosed as untraversable, as they have great slopes. In Fig.8(c), the edges of the stairs look more traversable after filtered by the "wheel dropping model.

V. Real-World Experiments and Results

A. System Architecture

We did experiments on the real robot. The system architecture and the results are shown in this section.

1) Hardware: The robot platform we use is designed by our lab, which is small, light and highly maneuverable. More details about the outlook about the robot could be seen in Fig.1(a).

The robot's motion pattern is special, which is like a four wheels steering car, so that it can drive sideways. The robot has four motors, two for throttle and two for steering. The movement of the robot is control by three signals: throttle, steering and the motion pattern, which determines whether the front and back wheels move in the same direction, in opposite direction, or which end of wheels steering. Because it is highly maneuverable and has different motion patterns than the general ground mobile robots, such as car-like robots and differential-driven robots, its traversability analysis and motion planning will be different from typical ones.

The robot has two computing resources, one is a Jetson TX2 board, which is responsible for main jobs of the system, such as processing camera data, and analyze terrain traversability. The other one is a Raspberry Pi board, which controls motors by pulse width modulation(PWM) through pins on the board. Besides, a WIFI router is mounted on the robot in order to provide a local network for the communication between different processing nodes.

For the sensor, the robot uses one ZED stereo camera on the top front, facing forward and downward. In this direction, although the camera has a smaller field of view compared to facing forward, it collects denser point cloud data.

The robot has lots of electronic components driven on various voltage. In order to make the hardware system compact,the robot's circuit uses many voltage regulators, therefor, it can only use one 7.4V battery for power supply. For details of hardware configuration, refer to Fig.1(c).

2) Software: In this project, we use ROS(robotic operating system) as a software platform, as it provides convenient ways for different processes to cooperate. In ROS, each node is a process that could run independently. This system has many nodes performing different functions, but it is mainly composed of three parts: sensor driving, traversability analysis, and motion control. The details of the software architecture could be seen in Fig.1(b).

- sensor driving: The stereo camera driving node provides 3D point cloud data, the robots position estimate, and the transforms between different coordinate systems(tf information in ROS). The robot does not have an IMU, but it uses computer vision algorithms to track the robot's position changing.
- Traversability analysis part consists of two nodes, one is for 2D map generation [5] [4], which converts 3D point cloud data to 2D elevation map; the other one is for traversability analysis. The pipeline for traversability analysis is shown in Fig.9. Firstly, smooth the elevation map by the "wheel dropping" model, then extracting geometry features, such as roughness and slopes, then calculate the traversability according to the geometry features as well as the robot's capability. Finally convert the traversability map to grid map for navigation.
- For motion control parts, the most basic process is

to transfer steering and throttle messages to voltage outputs. The robot could either be controlled by a joystick, which needs a node to map joystick message to steering and throttle message; or it could be controlled by navigation planners, which will be implemented in the future.

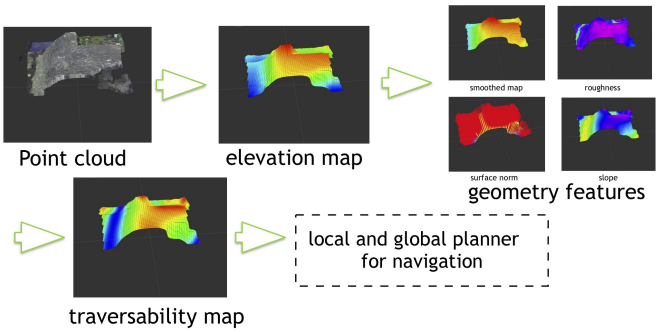
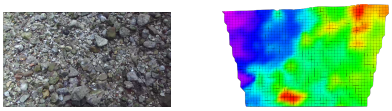


Fig. 9. Pipeline for the traversability analysis

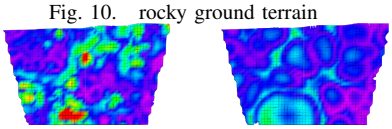
B. Real-World Results

The robot go outside recording some videos, then apply the traversability algorithm on the videos. Here are the results.



(a) image (b) elevation map

Fig. 10. rocky ground terrain

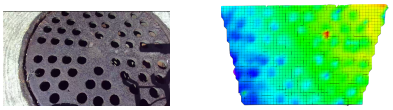


(a) without filter (b) with filter

Fig. 11. the traversability maps with and without the "wheel dropping" model

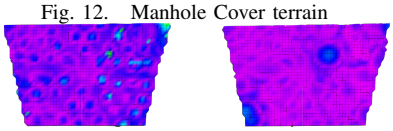
1) Rocky Ground: As shown in Fig.10, the ground is covered many stones, and looks rough, but as the stones are small, the robot could traverse on it easily. Without the "wheel dropping" model, there are some hot spots that looks not so traversable, see Fig.11(a), but with the filter, see Fig.11(b), the hot spots disappeared and the surface looks more traversable.

2) Manhole Cover: Then manhole cover has many small holes, as seen in Fig.12. The holes are very small compared to the robot's wheel, and have no influence on the robot's traverse. Without the "wheel dropping" model filter, the traversability of the hole looks different from the that of the surroundings, see Fig.13(a). However, with the filter, as seen in Fig.13(b), the holes do not look obvious, and the traversability map is smoother.



(a) image (b) elevation map

Fig. 12. Manhole Cover terrain



(a) without filter (b) with filter

Fig. 13. the traversability maps with and without the "wheel dropping" model

VI. Future Work

The future work mainly includes two parts, the first is to refine the traversability calculation formula. The traversability depends on both geometric features and the robot's capabilities. Now, the traversability is calculated by a weighted sum of roughness and slopes, and the parameter are manually set. There needs more work to refine the formula's form and parameters.

To really implement autonomous off-trial navigation, local and global planners need to be implemented. The robot will use GPS for localization and goal position setting, which are used for global planning. Traversability maps are converted to grid maps by setting an upper and lower threshold on traversability, which can be used for local planning. All points whose traversability are greater than the upper threshold are considered as free space, and all points that are smaller than the lower threshold are considered untraversable. For points between the lower threshold and upper threshold, they are traversable, but need more effort, for example, more power would be applied to the motors when executing a motion. Furthermore, the robot has different motion patterns from traditional car-like or differential-driven robots. The front wheels and back wheels are controlled separately, but the wheels on the same ends are controlled together. Although a car-like local planner could be used for this robot, to optimize its mobility, a new local planner that satisfies the robot is required.

Once these planners are complicated, the robot could autonomously navigate from one point to the other point in uneven outdoor terrains. This system could be applied to a lot of applications, such as oil exploration. People specify where they want to explore oil indoor and put the robots outdoor to autonomously explore, which can reduce the number of people sent to the field.

VII. Conclusion

Traversability analysis is crucial for robots working in unstructured outdoor environments to avoid collision and traver efficiently. Negative objects, such as holes and grooves need to pay attention in uneven terrains. Some negative objects are small compared to the robot's wheel, and will not cause hazards to the robot. They should be diagnosed as traversable even they have great height difference and

slopes. In this paper, we propose a "wheel dropping" model, which considering the actual location of the robot's wheel as it would sit on the ground. In this way, small negative objects are 'filled', which allows the robot to traverse over them that were falsely identified as untraversable by previous traversability analysis methods.

VIII. Acknowledgement

Thanks the members of the Robomechanics Lab at Carnegie Mellon University for their help on the robot's hardware and discussions.

References

[1] Motilal Agrawal, Kurt Konolige, and Robert C Bolles. Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach. In *2007 IEEE Workshop on Applications of Computer Vision (WACV'07)*, pages 7–7. IEEE, 2007.

[2] Igor Bogoslavskyi, Olga Vysotska, Jacopo Serafin, Giorgio Grisetti, and Cyrill Stachniss. Efficient traversability analysis for mobile robots using the kinect sensor. In *2013 European Conference on Mobile Robots*, pages 158–163. IEEE, 2013.

[3] Geert De Cubber, Daniela Doroftei, Hichem Sahli, and Yvan Baudoin. Outdoor terrain traversability analysis for robot navigation using a time-of-flight camera. In *Proc. RGB-D Workshop on 3D Perception in Robotics, Vasteras, Sweden*, 2011.

[4] Péter Fankhauser, Michael Bloesch, Christian Gehring, Marco Hutter, and Roland Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *International Conference on Climbing and Walking Robots (CLAWAR)*, 2014.

[5] Péter Fankhauser, Michael Bloesch, and Marco Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.

[6] Péter Fankhauser and Marco Hutter. A universal grid map library: Implementation and use case for rough terrain navigation. In *Robot Operating System (ROS)*, pages 99–120. Springer, 2016.

[7] Yoichi Morales, Alexander Carballo, Eijiro Takeuchi, Atsushi Aburadani, and Takashi Tsubouchi. Autonomous robot navigation in outdoor cluttered pedestrian walkways. *Journal of Field Robotics*, 26(8):609–635, 2009.

[8] Arnab Sinha and Panagiotis Papadakis. Mind the gap: Detection and traversability analysis of terrain gaps using lidar for safe robot navigation. *Robotica*, 31(7):1085–1101, 2013.

Following Social Groups: Socially Compliant Autonomous Navigation in Dense Crowds

Xinjie Yao[1], Ji Zhang[2] and Jean Oh[2]

***Abstract*— In densely populated environments, socially compliant navigation is critical for autonomous robots as driving close to people is unavoidable. This manner of social navigation is challenging given the constraints of human comfort and social rules. Traditional methods based on hand-craft cost functions to achieve this task have difficulties to operate in the complex real world. Other learning-based approaches fail to address the naturalness aspect from the perspective of collective formation behaviors. We present an autonomous navigation system capable of operating in dense crowds and utilizing information of social groups. The underlying system incorporates a deep neural network to track social groups and join the flow of a social group in facilitating the navigation. A collision avoidance layer in the system further ensures navigation safety. In experiments, our method generates socially compliant behaviors as state-of-the-art methods. More importantly, the system is capable of navigating safely in a densely populated area (10+ people in a $10\,\mathrm{m} \times 20\,\mathrm{m}$ area) following crowd flows to reach the goal.**

I. Introduction

The ability to safely navigate in populated scenes, e.g. airports, shopping malls, and social events, is essential for autonomous robots. The difficulty comes from the fact that people walk closely to the robot cutting ways in front of the robot or between the robot and the goal point. The safety margin for the robot to drive in crowded scenes is pushed to the minimum. In such a case, the navigation system has to trade-off between driving safely close to people and reaching the goal quickly. Further, previous study of socially compliant navigation [1] states three aspects in terms of the robot behaviors – *comfort* as the absence of annoyance and stress for humans in interaction with robots, *naturalness* as the similarity between the robot and human behaviors, and *sociability* as to abide by general cultural conventions. Among these three aspects, the first aspect essentially reflects safety of the navigation.

Previous studies on socially compliant navigation attempt to solve the problem with various methods, including data-driven approaches for human trajectory prediction [2], [3], potential field-based [4] and social force model-based [5] approaches. In particular, reinforcement learning-based methods use reward functions to penalizes improper robot behaviors eliminating the cause of discomfort [6], [7]. Inverse reinforcement learning based-methods learn from expert demonstrations [8]. These methods are hard to generalize due to that a large set of comprehensive expert demonstrations are hard to acquire.

The study of this paper is based on our previous work which uses deep learning in solving the socially compliant navigation problem [9]. This paper extends the work in two ways. First, we consider the findings from a previous study [10] that 70% of people walk in social groups. Crowd behavior can be summarized as flows of social groups, and humans tend to move along the flow. It is our understanding that the behavior of joining the flow that shares similar heading direction is more socially compliant, causing fewer collisions and disturbances to surrounding pedestrians. Our method recognizes social groups and selects the flow to follow. Second, we ensure safety with a multi-layer navigation system. In this system, a deep learning-based global planning layer makes high-level socially compliant behavioral decisions while a geometry-based local planning layer handles collision avoidance at a low-level.

The paper is further related to previous work on modeling aggregate interactions among social groups [10] and leveraging learned social relations in tracking group formations [11]. Our main contributions are a deep learning-based method for socially compliant navigation with an emphasis on tracking and joining the crowd flow and an overall system integrated with the deep leaning method capable of safe autonomous navigation in dense crowds.

II. Method

A. System Overview

Fig. 1 gives an overview of the autonomous navigation system which consists of three subsystems as follows.

- **State Estimation Subsystem** involves a multi-layer data processing pipeline which leverages lidar, vision, and inertial sensing [12]. The subsystem computes the 6-DOF pose of the vehicle as well as registers laser scan data with the computed pose.
- **Local Planning Subsystem** is a low-level planning subsystem in charge of obstacle avoidance in the vicinity of

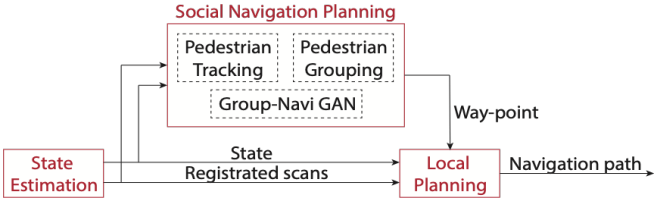


Fig. 1: Navigation software system diagram.

[1]X. Yao is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology. Email: xyaoab@ust.hk

[2]J. Zhang and J. Oh are with the Robotics Institute, Carnegie Mellon University. Emails: zhangji@cmu.edu, jeanoh@cmu.edu

the vehicle. The planning algorithm involves a trajectory library and computes collision-free paths for the vehicle to navigate [13].

- **Social Navigation Planning Subsystem** takes in observations only consisting of pedestrians by subtracting the prior map. The subsystem tracks pedestrians in the surroundings of the vehicle, and then extracts the grouping information from the pedestrian walking patterns, with which, the subsystem generates way-points (as input of the Local Planning Subsystem), leveraging Group-Navi GAN, a generative planning algorithm in an adversarial training framework based on a deep neural network, Navi-GAN [9].

B. Group-Navi GAN

Following the extended social force model [10], we propose Group-Navi GAN, a framework to jointly address the safety and naturalness aspects at a group's level. Group-Navi GAN is inspired by our previous work Navi-GAN [9] which models social forces at an individual's level. An intention-force generator in the Group-Navi GAN deep network models the driving force as $\overrightarrow{f_i}^0$ for target agent i to move toward the goal. A group-force generator models the repulsive force from other pedestrians j as $\overrightarrow{f_{ij}}$ and the interaction force from other group members as $\overrightarrow{f_i}^{group}$. The joint output of the intention-force generator and group-force generator defines the path for the robot to navigate.

In the group-force generator, a group pooling module first associates the target agent to a group based on the motion information (see Fig. 2). Then, the group pooling module computes path adjustments which essentially guide the robot to follow the group. We apply a support vector machine classifier [14] trained by [11] to determine if two agents belong to the same group. This uses the local spatio-temporal relation to cluster the agents with similar motions based on the coherent motion indicators, i.e. the differences in walking speed, spatial locations, and headings.

We use the following equation to aggregate the hidden state from h_j^t to ${h'_j}^t$,

$$ {h'_j}^t = I_{ij}[s_i = s_j] * cos(\theta_i - \theta_j) * h_j^t, \tag{1} $$

where $I_{ij}[s_i = s_j]$ indicates if two agents are in the same group,

$$ I_{ij}[s_i = s_j] = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same group} \\ 0, & \text{otherwise} \end{cases} \tag{2} $$

θ_i and θ_j are the agent headings. The resulting embedding H_i^t of hidden state ${h'_j}^t$ is computed as a row vector which consists of the maximum elements from all other agents. The embedding is further concatenated for decoding,

$$ {H'_i}^t = [H_i^t, h_i^t, n_i] \tag{3} $$

where n_i is random noise drawn from $\mathcal{N}(0, 1)$.

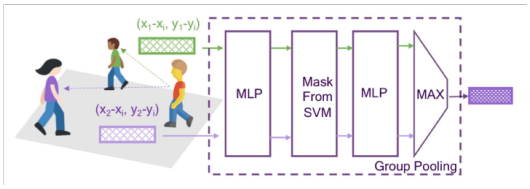


Fig. 2: Group pooling module in the Group-Navi GAN deep network. The input of the module is the relative displacements of the surrounding pedestrians w.r.t. the target agent. The module associates the target agent to a group based on the motion information and outputs path adjustments for the robot to follow the group.

III. Experiments

A. Social Compliance Evaluation

We evaluate our method on two publicly available datasets: ETH [15] and UCY [16]. These datasets include rich social interactions in real-world scenarios. We follow the same evaluation methodology as the leave-one-out approach and the error metrics used in the prior work [3]:

1) *Average Displacement Error*: The average L2 distance between predicted way-points and ground-truth trajectories over the predicted time steps.
2) *Final Displacement Error*: The L2 distance between the predicted way-point and true final position at the last predicted time step.

We compare against a linear regressor that only predicts straight paths, Social-GAN(SGAN) [3], and Navi-GAN [9]. We use the past eight time steps to predict the future eight time steps. As shown in TABLE I, our method yields considerable accuracy improvements for some of the datasets where rich group interactions are prevalent. In particular, UNIV and ZARA1 have more than 70% of the pedestrians moving in social groups, and thus our model performs better. Our model performs slightly worse than the state-of-the-art approaches with the ETH and HOTEL datasets due to the lack of social group interactions. Further, our method assumes the existence of a goal point for each person in the dataset. Lacking precise goal point information results in a relative low accuracy. In the next experiments, we will show results with author-collected data where the strength of our method is more obvious.

Metric	Dataset	Group Percentage	Linear	SGAN [3]	Navi-GAN [9]	Group-Navi GAN
ADE	**ETH** [15]	18%	0.84	**0.60**	0.95	1.33
	HOTEL [15]	19%	**0.35**	0.48	0.43	0.39
	UNIV [16]	**73%**	0.56	0.36	0.85	**0.29**
	ZARA1 [16]	**70%**	0.41	0.21	0.40	**0.21**
	ZARA2 [16]	69%	0.53	**0.27**	0.47	0.30
	AVG	50%	0.54	0.39	0.62	0.50
FDE	**ETH** [15]	18%	1.60	**1.22**	1.64	1.98
	HOTEL [15]	19%	**0.60**	0.95	0.74	0.93
	UNIV [16]	**73%**	1.01	0.75	1.36	**0.68**
	ZARA1 [16]	**70%**	0.74	0.42	0.66	**0.40**
	ZARA2 [16]	69%	0.95	**0.54**	0.72	0.85
	AVG	50%	0.98	0.78	1.02	0.96

TABLE I: Social compliance evaluation of Group-Navi GAN and other baseline approaches. Two error metrics, Average Displacement Error and Final Displacement Error are reported (in meters) for $t_{obs} = 8$ and $t_{pred} = 8$. We manually count the number of pedestrians moving in social groups. Our method outperforms the prior work with the UNIV and ZARA1 datasets where social groups are richly available.

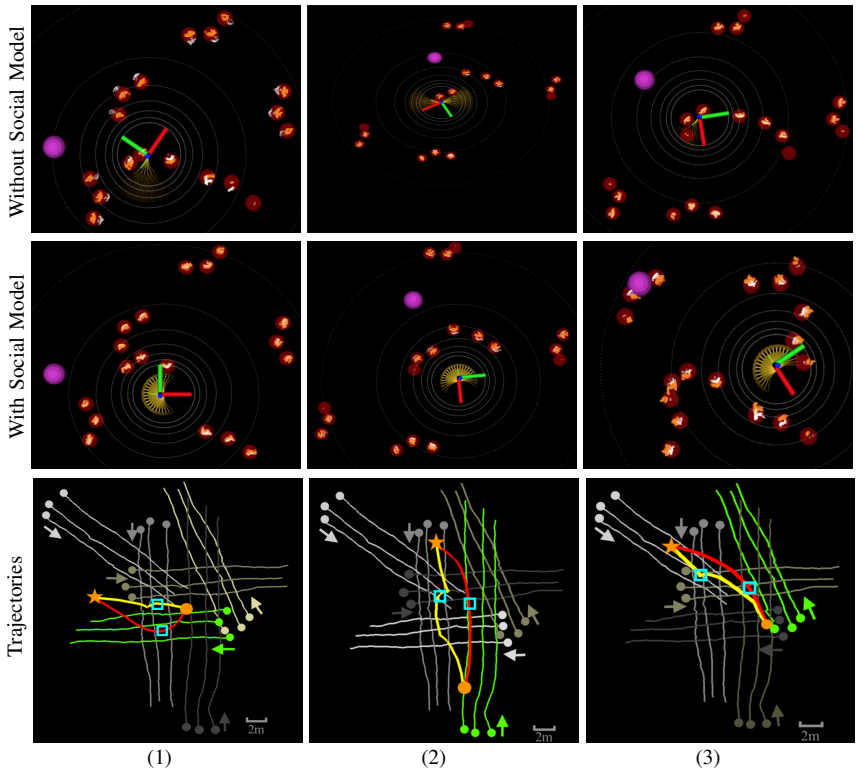


Fig. 4: Simulation results in a $10\,\mathrm{m} \times 20\,\mathrm{m}$ area. The tests involve 18 people walking in 6 groups. Each group moves in a different direction. The three columns present three representative cases. The first and second rows show screenshots of the simulation environment. The coordinate frame indicates the robot. The goal point is marked as the magenta dot. The red dots are the tracked pedestrians using laser scan data. The third row displays the trajectories of the pedestrians (gray and green) and the robot (yellow and red). The dots are the start points and the star is the goal point of the robot. When using Navigation without Social Model, the robot produces the yellow path. When using Navigation with Social Model, the robot follows the group in green color and produces the red path. A blue square is labeled on each robot path where the corresponding screenshot is captured on the first and second rows. Specifically, on the first row, the screenshots show the moments when the robot drives overly close to people due to not using the social model. On the second row, the screenshots are taken while the robot follows a group during the navigation.

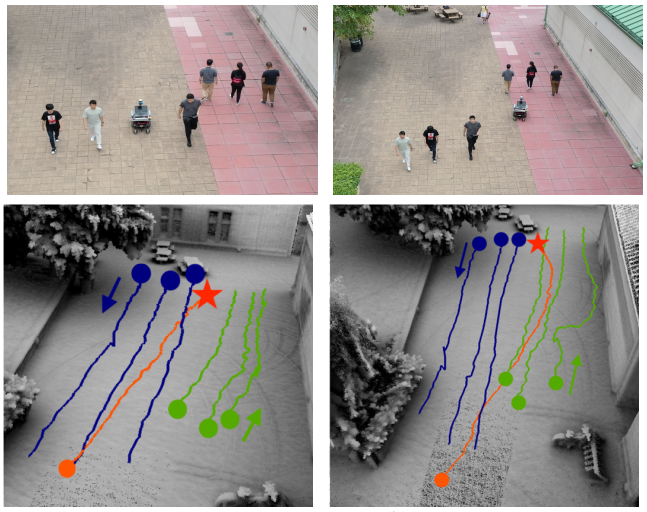
B. Group Following Evaluation

We further evaluate the method with a robot vehicle as shown in Fig. 3. The robot is equipped with a Velodyne Puck laser scanner for collision avoidance and pedestrian tracking. Our method is evaluated in two configurations – Navigation with Social Model refers to the full navigation system as shown in Fig. 1, and Navigation without Social Model has the Social Navigation Planning Subsystem removed. The State Estimation Subsystem and the Local Planning Subsystem are directly coupled. The robot navigates directly toward the goal and uses the Local Planning Subsystem to avoid collisions locally.

We show results in both simulation and real-work experiments with pedestrian data collected by the robot. In simulation, we show scenarios with 18 people walking around the robot in 6 groups. In real-work experiments, we have 6 people walking in 2 groups. One group moves along the robot navigation direction and the other group moves in the opposite direction. The results are shown in Fig. 4 and Fig. 5. In each scenario, the robot selects a group to follow with the full navigation system (Navigation with Social Model). If using Navigation without Social Model, the robot drives directly toward the goal and results in interactions with groups moving in other directions.

Finally, we conduct an Amazon Mechanical Turk (AMT) study to further understand the safety and naturalness of the robot navigation. A total of 466 participants evaluate the simulation and real-world results. As shown in Table II, $>$ 90% of the participants consider Navigation without Social Model to be unsafe (with collisions) while the ratio reduces to $<$ 40% using Navigation with Social Model. With the

Fig. 3: Experiment platform. A wheelchair-based robot carries a sensor pack on the top. The sensor pack consists of a Velodyne Puck laser scanner, a camera, and a low-grade IMU. The scan data is used for collision avoidance and pedestrian tracking. A laptop computer carries out all onboard processing.



(a) Without Social Model (b) With Social Model

Fig. 5: Real-world experiments in a $10\,\mathrm{m} \times 20\,\mathrm{m}$ area. The first row shows photos of 6 people walking in 2 groups. One group moves along the robot navigation direction and the other group moves in the opposite direction. The second row shows the corresponding trajectories of the people (blue and green) and the robot (orange). Dots indicate the start points and the star indicates the goal point of the robot. In (a), when using Navigation without Social Model, the robot drives directly toward the goal point and results in cutting through the group on the left that moves against the robot. In (b), when using Navigation with Social Model, the robot follows the group on the right and avoids disturbances to the pedestrians.

real-world results, 95% of the participants report that the robot forces other pedestrians to change their paths if using Navigation without Social Model. When using Navigation with Social Model, the ratio reduces to 4%. The survey result validates that our method helps reduce disturbances to other pedestrians as well as improves safety of the navigation. A video of these results can be seen at www.youtube.com/watch?v=I_SkA9rmxYE.

IV. Conclusion

The paper proposes an autonomous navigation system capable of operating in dense crowds. In this system, a Social Navigation Planning Subsystem incorporating a deep neural network generates socially compliant behaviors. This involves a group pooling mechanism by inferring social relationships to encourage the autonomous navigation to join the flow of a social group sharing the same moving direction. We show the effectiveness of our method through quantitative and empirical studies in both simulations and real-world experiments. The result is that by joining the crowd flow, the robot has fewer collisions with people crossing sideways or walking toward the robot. Joining the flow also creates fewer disturbances to the pedestrians. As a result, the robot navigates in a safe and natural manner. Since this paper focuses on human-robot interactions at a group's level, extension of the work in the future can model interactions between groups and scattered individuals.

Metric	Scene	Without Social Model	With Social Model
Collision (Safety)	(1)	97%	42%
	(2)	92%	6%
	(3)	92%	36%
	AVG	93%	28%
Path Change (Naturalness)	Real world	95%	4%

TABLE II: Results of survey study. A total of 466 participants evaluate the simulation results in Fig. 4 and the real-world results in Fig. 5. We can see that $> 90\%$ of the participants consider the Navigation without Social Model to have collisions. For Navigation with Social Model, the ratio reduces to $< 40\%$. Further, 95% of the participants report that the robot forces other pedestrians to change their paths if using Navigation without Social Model. When using Navigation with Social Model, the ratio reduces to 4%. The ratios reduce by 3 times in terms of collision and 20 times in terms of path change which validate that our method helps reduce disturbances to other pedestrians as well as improves safety.

Acknowledgment

Special thanks are given to C.-E. Tsai, Y. Song, D. Zhao for facilitating experiments.

References

[1] T. Krusea, A. K. Pandeybc, R. Alamibc, and A. Kirschd, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, Dec 2013.

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F.-F. Li, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.

[3] G. Agrim, J. Justin, F.-F. Li, S. Silvio, and A. Alexandre, "Social gan: Socially acceptable trajectories with generative adversarial networks," *arXiv preprint arXiv:1803.10892*, 2018.

[4] F. Hoeller, D. Schulz, M. Moors, and E. F. Schneider, "Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[5] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, 1998.

[6] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *arXiv preprint arXiv:1703.08862*, 2018.

[7] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *arXiv preprint arXiv:1809.08835*, 2019.

[8] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot naviga- tion via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, 2016.

[9] C.-E. Tsai, "A generative approach for socially compliant navigation," Master's thesis, Pittsburgh, PA, June 2019.

[10] M. Moussad, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *arXiv preprint arXiv:1003.3894*, 2010.

[11] T. Linder and K. O. Arras, "Multi-model hypothesis tracking of groups of people in rgb-d data," in *Proc.of the 17th Intl Conf. on Information Fusion*, 2014.

[12] J. Zhang and S. Singh, "Random field topic model for semantic region analysis in crowded scenes from tracklets," in *Journal of Field Robotics*, vol. 35, no. 8, 2018.

[13] J. Zhang, C. Hu, R. Gupta Chadha, and S. Singh, "Maximum likelihood path planning for fast aerial maneuvers and collision avoidance," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[14] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, no. 20, 1995.

[15] S. Pellegrini, A. Ess, and L. V. Gool, "Improving data association by joint modeling of pedestrian trajectories and groupings," in *Computer VisionECCV*, 2010.

[16] L. Leal-Taixe, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Improving data association by joint modeling of pedestrian trajectories and groupings," in *CVPR*, 2014.

Testing 3D Object Detection Methods with Visualization of Point Clouds and Bounding Boxes

Chengwei Ye

August 16, 2019

1 Introduction

Object detection plays a significant role in the task of perception in autonomous driving, and lays the foundation for subsequent tasks of path planning, decision making, and vehicle control. Some state-of-the-art methods to perform 3D object detection over LIDAR point clouds utilize 2D CNN or 3D CNN after bird's-eye-view projection or voxelization [1] [2] [3] [4], while others directly generate 3D box proposals from point clouds in a bottom-up manner. However, these methods may potentially suffer from the problem of overfitting, where the performance exhibits large variations given changes in the datasets used or the scenes displayed.

PointRCNN is a two-stage 3D object detector proposed by Shaoshuai Shi *et al.* The framework performs 3D object detection from raw point clouds in two stages: in stage-1, high-quality 3D proposals are generated in a bottom-up manner by segmenting the point clouds into foreground points and background points; in stage-2, the proposals are refined in canonical coordinates by combining local spatial features and global semantic features to obtain final detection results [5]. The authors claim that the PointRCNN architecture outperforms state-of-the-art methods with remarkable margins when tested on the 3D detection benchmark of KITTI dataset.

We first evaluate PointRCNN on the KITTI dataset to verify whether the performance of the pre-trained model matches the accuracy claimed by the author. We notice that the running outputs of PointRCNN contain only accuracy data given by comparison between predicted results and known benchmarks, but the LIDAR data scanned by our lab have not been labeled. Therefore, we visualize the velodyne point clouds and predicted bounding boxes in both 3D space and x-y plane using MATLAB to check whether the bounding boxes mark the vehicles accurately. Finally, we standardize the data structure of our data to be consistent with KITTI velodyne data, and test PointRCNN on our dataset.

This paper is structured as follows: Section Two briefly introduces the structure and usage of the KITTI 3D object detection dataset, and records steps to set up the virtual environment for PointRCNN, prepare data, and evaluate the pre-trained model on the KITTI dataset. Section Three discusses the details of using MATLAB to visualize 3D point clouds and 3D bounding boxes separately, integrate them together in Velodyne coordinates, and project them from 3D space to the x-y plane. Section Four focuses on converting the data format and structure of our data into the same as that of the KITTI velodyne data, and modifying data split settings in PointRCNN configuration files, in order to test the PointRCNN pre-trained model on our data. Section Five discusses the possible directions and expected outcomes of future work.

2 Testing PointRCNN on KITTI Dataset

2.1 Environment Set-up

First, we build an Anaconda virtual environment for testing PointRCNN. Since PointRCNN requires the Python version to be 3.6 or above, we create a virtual environment and activate it with the following lines:

```
conda create -n prcnn_env python=3.6
activate prcnn_env
```

New packages will be installed under virtual environment prcnn_env from now on, so that the running environment for other modules on the server will not be affected. PointRCNN also requires PyTorch with version 1.0, so we install PyTorch on virtual environment prcnn_env with the following line. Note that our CUDA version is 8, for other CUDA versions, 'cuda80' should be changed into the corresponding version.

```
conda install pytorch=1.0 cuda80 -c pytorch
```

PointRCNN also requires dependent python libraries including easydict, tqdm, and tensorboardX. These libraries can be installed with the following lines. Note that if error messages given by later steps indicate the lack of certain dependent libraries, they can be installed in a similar way.

```
conda install -c conda-forge easydict
conda install -c conda-forge tqdm
conda install -c conda-forge tensorboardx
```

Now the virtual environment prcnn_env is ready for installing PointRCNN. We download, build, and install PointRCNN with the following lines.

```
git clone --recursive https://github.com/sshaoshuai/PointRCNN.git
sh build_and_install.sh
```

2.2 Dataset Preparation

The 3D object detection benchmark in the KITTI dataset consists of 7481 training point clouds and 7518 testing point clouds [6]. Eight different classes are labeled in the benchmark, but only the classes 'Car' and 'Pedestrian' are evaluated. The sub-folder named velodyne/ contains the 3D laser scan data in bin format; the sub-folder named image_2/ contains the left color camera images in png format; the sub-folder named label_2/ contains the left color camera label files in txt format; the sub-folder named calib/ contains the calibration for all four cameras in txt format. The data structures in velodyne and label files will be introduced in future sections.

The datasets can be downloaded from http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d using the 'wget' command. The downloaded sub-folders calib/, velodyne/, label_2/, and image_2/ must be arranged into the structure shown in the following image, for the purpose of valid data retrieval.

```
PointRCNN
├── data
│   ├── KITTI
│   │   ├── ImageSets
│   │   ├── object
│   │   │   ├──training
│   │   │   │   ├──calib & velodyne & label_2 & image_2
│   │   │   ├──testing
│   │   │   │   ├──calib & velodyne & image_2
├── lib
├── pointnet2_lib
├── tools
```

Figure 1: KITTI dataset sub-folders structure.

2.3 Evaluating the Pre-trained Model

The pre-trained model of PointRCNN is trained on the 'train' data split with 3712 samples. We evaluate the model on the 'val' data split with 3769 samples with the following line:

```
python eval_rcnn.py
--cfg_file cfgs/default.yaml
--ckpt PointRCNN.pth
--batch_size 1
--eval_mode rcnn
--set RPN.LOC_XZ_FINE False
```

The evaluation module is borrowed from https://github.com/traveller59/kitti-object-eval-python, which performs fast KITTI object detection evaluation in python and supports 2d, bev, 3d, and aos. The evaluation metric is Average Precision(AP) with IoU threshold set to 0.7. The performance of the pre-trained model on the 'val' data split is shown as follows:

	AP(IoU=0.7)		
	Easy	Moderate	Hard
bbox AP	96.91	89.53	88.74
bev AP	90.21	87.89	85.51
3d AP	89.19	78.85	77.91
aos AP	96.90	89.41	88.54

Figure 2: Performance of the pre-trained model on the Car class of KITTI 'val' data split. The evaluation metric is Average Precision(AP) with IoU threshold 0.7. For each of 2d evaluation, bev evaluation, 3d evaluation, and aos evaluation, the three precision numbers correspond to easy, moderate, and hard respectively.

3 Visualization of Point Clouds and Bounding Boxes

3.1 3D Visualization of Point Clouds

PointRCNN utilizes the velodyne 3D laser scan data in the KITTI dataset. The scans of velodyne point clouds are stored as an $N \times 4$ float matrix into binary files with names starting from 000000.bin. In the matrix, each row contains four float values: the first three values correspond to x, y, and z coordinates of a specific point, while the last value corresponds to the reflectance information of this point. Since the scans are stored in a row-aligned manner, four consecutive float values correspond to one measurement.

We utilize MATLAB for visualization of the velodyne point clouds. Functions fopen() and fread() are employed to read the binary file into a $N \times 1$ float matrix. Note that it's essential to specify the data type to be float when invoking the fread() function. Based on the data structure of velodyne data, a simple re-shape of the matrix is performed after retrieving the float values, converting the $N \times 1$ matrix into a $\frac{N}{4} \times 4$ matrix.

For each row in the $\frac{N}{4} \times 4$ matrix, the first, second, and third entries correspond to the x, y, and z coordinates of a point respectively. The scatter3() function is then employed to plot the points in 3D space. Among the three images below, the left one shows the $N \times 1$ matrix obtained by fread(), where only the first four measurements are included; the middle one shows the converted $\frac{N}{4} \times 4$ matrix; and the right one shows the plot of the overall point cloud.

463904x1 double

	1	2	3	4
1	57.6570			
2	20.9960			
3	2.2840			
4	0			
5	57.6020			
6	21.1810			
7	2.2840			
8	0			
9	57.3440			
10	21.7020			
11	2.2820			
12	0			
13	57.0520			
14	21.6940			
15	2.2730			
16	0			

115976x4 double

	1	2	3	4
1	57.6570	20.9960	2.2840	0
2	57.6020	21.1810	2.2840	0
3	57.3440	21.7020	2.2820	0
4	57.0520	21.6940	2.2730	0

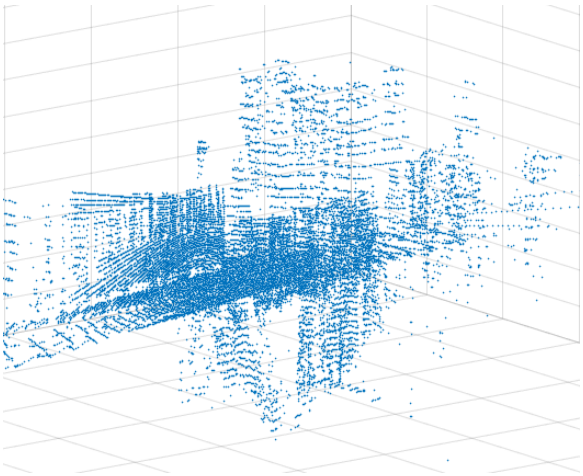


Figure 3: Visualization of 3D point clouds.

000001.txt

```
Car -1 -1 -1.7723 753.6551 163.8756 814.0420 204.0526 1.5462 1.6426 3.9845 7.0502 1.2010 29.7814 -1.5398 1.6346
Car -1 -1 -1.3333 392.5066 183.8779 423.7497 203.3627 1.4735 1.5792 3.8999 -16.4254 2.4020 58.7729 -1.6059 1.2250
Car -1 -1 2.0431 251.0295 185.3513 298.6565 207.6607 1.4398 1.5592 3.5853 -23.3947 2.3451 50.4248 1.6087 0.3313
Car -1 -1 -1.6898 673.9335 173.1354 726.0756 212.4593 1.4764 1.6120 3.9510 3.5157 1.4889 29.0915 -1.5695 -0.8158
```

Figure 4: Data structure of result file 000001.txt.

3.2 3D Visualization of Bounding Boxes

PointRCNN performs object detection in a frame-by-frame manner. For each frame, the prediction results of bounding boxes are stored in a txt file, with file name following the form of 000001.txt. To understand the data structure of the result files, the data stored in 000001.txt are shown in the above image (Figure 4).

In the frame labeled as 000001, four objects are detected, each corresponding to one row in the file. In each row, the first value describes the type of the object; the ninth, tenth, and eleventh values describe the dimensions of the object, corresponding to its height, width, and length respectively; the twelfth, thirteenth, and fourteenth values describe the location of the object, corresponding to the x, y, and z coordinates of its center; the fifteenth value describes the rotation ry around the y-axis in camera coordinates.

With the information of each object's predicted dimensions, location, and rotation, we are able to calculate the coordinates of its eight corners, and subsequently plot the bounding boxes in 3D space. The MATLAB codes for calculating corner coordinates are as follows:

```
pos=[x,y,z]
x_c=[l/2,l/2,-l/2,-l/2,l/2,l/2,-l/2,-l/2]
y_c=[0,0,0,0,-h,-h,-h,-h]
z_c=[w/2,-w/2,-w/2,w/2,w/2,-w/2,-w/2,w/2]

R=[cos(ry)  0  sin(ry);
      0     1     0;
  -sin(ry)  0  cos(ry)]

c3d=[x_c;y_c;z_c]
c3d=(R*c3d)'
c3d=c3d+[pos;pos;pos;pos;pos;pos;pos;pos]
```

Here, c3d is a 8×3 matrix, where each row in the matrix corresponds to one corner of the bounding box, and the three values in one row correspond to the x, y, and z coordinates of a corner point. The plot3() function is then employed to connect the 8 corners and plot the 12 edges of the bounding box. The following two images illustrate the visualization of prediction results recorded in 000001.txt: the left one shows the c3d matrix of the fourth predicted object (corresponding to the fourth row in 000001.txt); the right one shows all bounding boxes plotted in 3D space.

corners3d

8x3 double

	1	2	3	4
1	2.7123	1.4889	31.0680	
2	4.3243	1.4889	31.0660	
3	4.3191	1.4889	27.1150	
4	2.7071	1.4889	27.1170	
5	2.7123	0.0125	31.0680	
6	4.3243	0.0125	31.0660	
7	4.3191	0.0125	27.1150	
8	2.7071	0.0125	27.1170	
9				

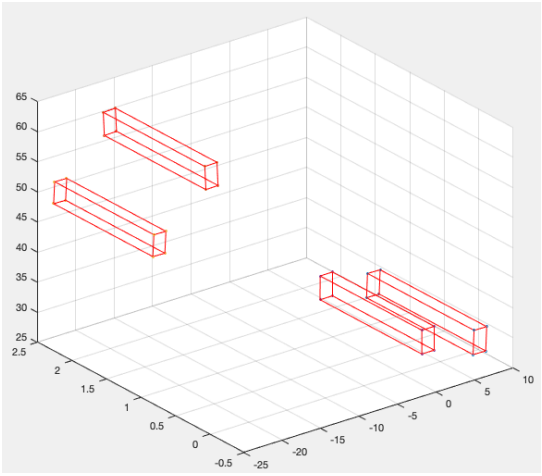


Figure 5: Visualization of 3D bounding boxes.

3.3 Integrating Point Clouds with Bounding Boxes

In the previous sections, we have implemented the visualization of point clouds and bounding boxes. In order to evaluate whether the predictions are accurate (i.e., whether the bounding boxes mark the cars well), point clouds and bounding boxes have to be integrated together. However, it is invalid to directly plot point clouds and bounding boxes into one plot, because their coordinates are not consistent.

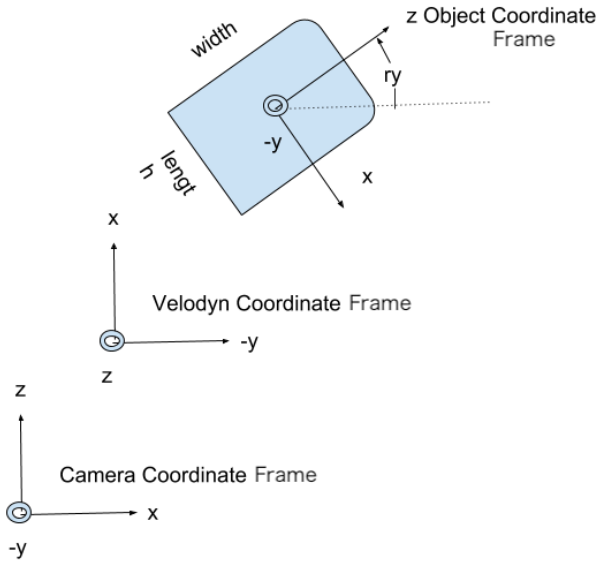


Figure 6: Relationship among Velodyne Coordinates, Camera Coordinates, and Object Coordinates.

The x, y, and z coordinates of point clouds are in Velodyne Coordinates, while the x, y, and z coordinates of bounding boxes are in Camera Coordinates. The conversion relationship between Velodyne Coordinates and Camera Coordinates is illustrated in the above image (Figure 6) [7]. We convert the x, y, and z coordinates of bounding boxes from Camera Coordinates to Velodyne Coordinates when calculating matrix c3d.

```
pos=[x,y,z]
x_c=[l/2,l/2,-l/2,-l/2,l/2,l/2,-l/2,-l/2]
y_c=[0,0,0,0,-h,-h,-h,-h]
z_c=[w/2,-w/2,-w/2,w/2,w/2,-w/2,-w/2,w/2]

R=[cos(ry)  0  sin(ry);
      0     1     0;
  -sin(ry)  0  cos(ry)]

c3d=[x_c;y_c;z_c]
c3d=(R*c3d)'
c3d=c3d+[pos;pos;pos;pos;pos;pos;pos;pos]

x_c=c3d(:,3)
y_c=c3d(:,1)*(-1)
z_c=c3d(:,2)*(-1)
c3d_vel = [x_c  y_c  z_c]
```

Here, matrix c3d_vel contains the locations of corners in Velodyne Coordinates. In order to focus on objects close to the camera, we set limitations on the x, y, and z coordinate range. The range of the x-axis is limited to [0, 60]; the range of the y-axis is limited to [-10, 20]; the range of the z-axis is limited to [-5, 2]. Points and bounding boxes which exceed the limitation will not be shown on the integrated plot.

Among the following images, the left one shows the point cloud of the frame labeled as 000001; the middle one shows the predicted bounding boxes of the frame labeled as 000001 in Camera Coordinates; the right one shows the integration of them in Velodyne Coordinates. Note that due to the limitation on the coordinate range, some boxes are missing or incomplete.

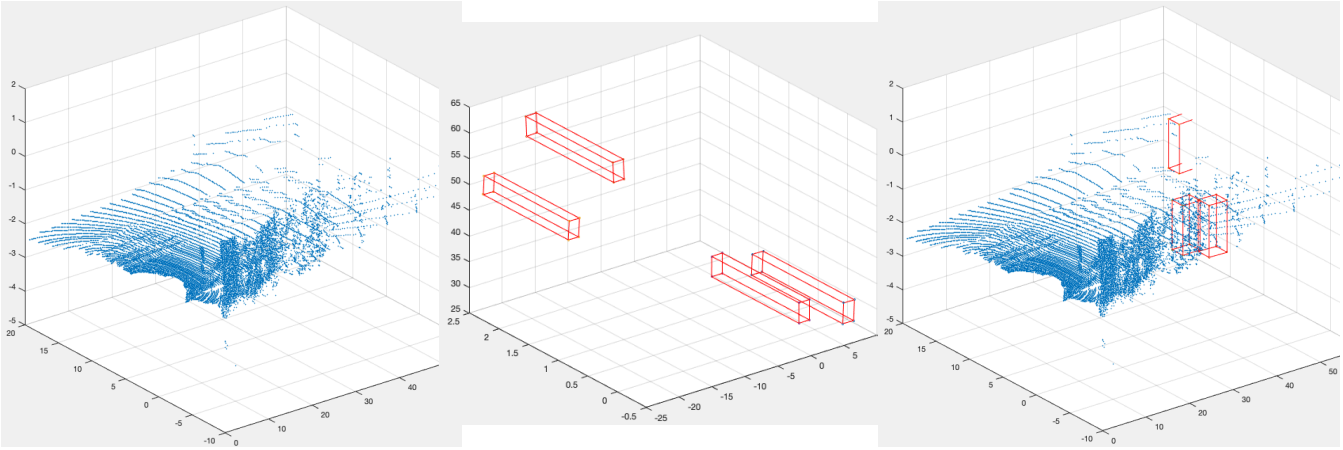


Figure 7: Integrating point clouds with bounding boxes.

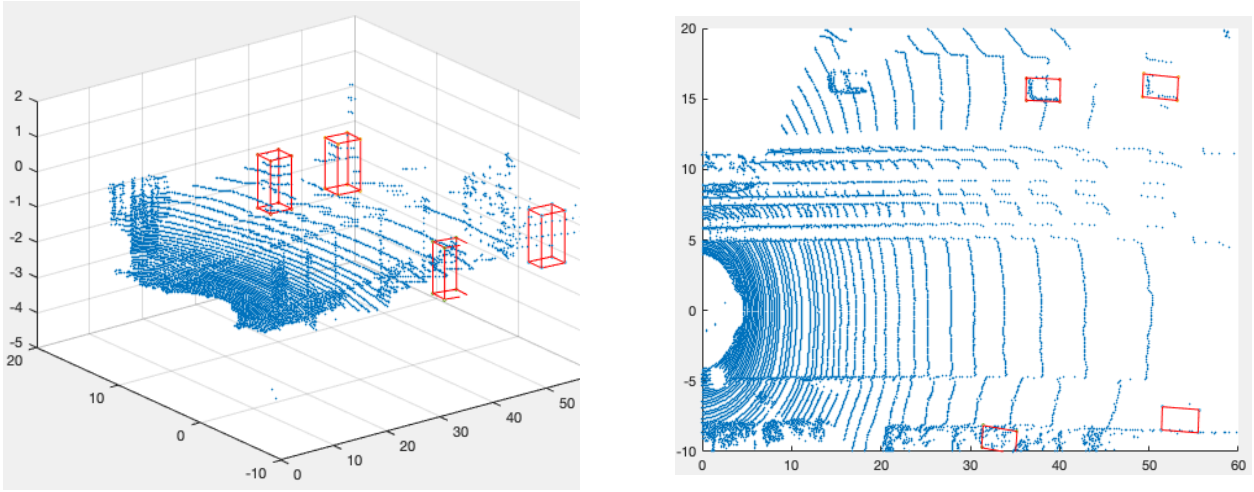


Figure 8: Projection of point clouds and bounding boxes from 3D space to x-y plane.

3.4 2D Visualization of Point Clouds and Bounding Boxes

Due to potential disturbance during the LIDAR scanning process and the complexity of point clouds, sometimes it is difficult to distinguish a vehicle from its background in the point clouds. We notice that a vehicle tends to exhibit an approximate L-shape when viewed from the top. Therefore, we project the point clouds together with predicted bounding boxes from 3D space to the x-y plane. In MATLAB, this projection can be performed by changing the scatter3() function into the scatter() function, and changing the plot3() function into the plot() function.

Figure 8 illustrates the projection from 3D space to the x-y plane: the left one shows the 3D point clouds and predicted bounding boxes of the frame labeled as 000004; the right one shows the 2D point clouds and predicted bounding rectangles of frame 000004.

4 Testing PointRCNN on Our Dataset

4.1 Standardizing Data Structure

In order to test PointRCNN on our own dataset, we first convert the data structure of our data into the same as that of the KITTI velodyne 3D laser scan data. Our data are stored in csv format, with each frame corresponding to one csv file. In the file, each row stores the scanned information of one point. The eighth, ninth, and tenth columns store x, y, and z coordinates of the point, which correspond to the first three columns in the KITTI velodyne data; the first column stores the reflectance information of this point, which corresponds to the last column in the KITTI velodyne data. The data structure of our data is shown in the following image.

intensity	laser_id	azimuth	elevation	distance_m	timestamp	vtkOriginalP	Points:0	Points:1	Points:2	__vtkIsSelected__
0	0	0.17	7.096	0	297000099	0	0	0	0	1
0	1	0.17	6.095	0	297000099	1	0	0	0	1
0	2	0.17	5.089	0	297000099	2	0	0	0	1
0	3	0.17	4.081	0	297000099	3	0	0	0	1
3	4	0.17	3.068	46.364	297000099	4	-1.91452	46.2579	2.48145	1
0	5	0.17	2.055	0	297000099	5	0	0	0	1
0	6	0.17	1.72	0	297000099	6	0	0	0	1
7	7	0.17	1.374	50.84	297000099	7	-4.3166	50.6417	1.21907	1
0	8	0.17	1.04	0	297000099	8	0	0	0	1
0	9	0.17	0.705	0	297000099	9	0	0	0	1
6	10	0.17	0.359	51.208	297000099	10	-4.34901	51.022	0.320854	1

Figure 9: Data structure of our data. In csv format.

We utilize MATLAB to perform the conversion of the data structure. The csvread() function is employed to read the raw data into a $N \times 11$ matrix. The first, eighth, ninth, and tenth columns are selected and arranged in order, forming a $N \times 4$ matrix. Since in KITTI velodyne data, the scans are stored in a row-aligned manner, the $N \times 4$ matrix has to be re-shaped into a $4N \times 1$ matrix, with four consecutive values corresponding to one measurement. Functions fopen() and fwrite() are then employed to write the matrix values into a binary file. Note that it's essential to specify the data type to be float when invoking the fwrite() function.

4.2 Testing the Pre-trained Model

After standardizing the data structure of our data, we replace the KITTI velodyne data in the /testing/velodyne/ directory with our data. When running the pre-trained model, the default data split is set to be the 'val' data split with 3769 samples. In order to test the model on our data, we modify two settings in the configuration files: 1. In config.py under /lib/ directory, the default value for __C.TEST.SPLIT is 'val', and we modify this value into 'test'; 2. In default.yaml under /tools/cfgs/ directory, the default value for SPLIT in TEST is 'val', and we modify this value into 'test'. The following two images show the places to modify.

```
__C.TEST = edict()
__C.TEST.SPLIT = 'test'
__C.TEST.RPN_PRE_NMS_TOP_N = 9000
__C.TEST.RPN_POST_NMS_TOP_N = 300
__C.TEST.RPN_NMS_THRESH = 0.7
__C.TEST.RPN_DISTANCE_BASED_PROPOSE = True
```

```
TEST:
    SPLIT: val
    RPN_PRE_NMS_TOP_N: 9000
    RPN_POST_NMS_TOP_N: 100
    RPN_NMS_THRESH: 0.8
    RPN_DISTANCE_BASED_PROPOSE: True
```

Figure 10: Modifying data split settings in configuration files config.py (above) and default.yaml (below).

Once the data split is set to 'test', data will be loaded according to the test.txt file in the /data/KITTI/ImageSets/ directory when running the pre-trained model. The original test.txt file contains numbers starting from 000000 up to 007517, which correspond to the names of KITTI velodyne testing files. Following this test.txt file, errors occur during the data loading process, because we have replaced the KITTI velodyne data in the /testing/velodyne/ directory with our own data. Our data only contain 850 samples, so the file names no longer match. Therefore, we update the test.txt file into a new one containing numbers starting from 000000 up to 000849, which correspond to the names of our data files.

Now we can test the PointRCNN pre-trained model on our data by running the following line:

```
python eval_rcnn.py
--cfg_file cfgs/default.yaml
--eval_mode rcnn
--test
```

The following image (Figure 11) shows the message printed out after executing the above test command. According to the message, the value of cfg.TEST.SPLIT has been set to 'test', and the total number of test samples is 850, indicating that our own data are loaded instead of the 7518 test samples in the KITTI velodyne dataset.

Due to the absence of benchmarks to compare with, tests of the pre-trained model on our dataset give no precision outputs similar to those shown in Figure 2. The tests performance can only be evaluated by visualizing the point clouds together with predicted bounding boxes and checking whether the bounding boxes mark the vehicles accurately, which will be part of the future work.

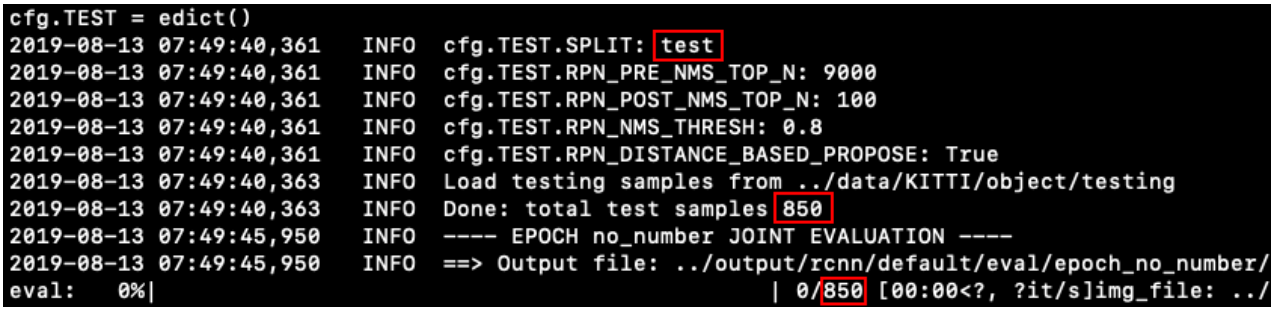


Figure 11: Testing PointRCNN pre-trained model on our dataset with 850 test samples.

5 Future Work

In the future, we will first identify situations where prediction accuracy of PointRCNN and other 3D object detection methods show a clear decrease, and will then improve upon them by modifying the network structures or introducing new ideas. We will also project the object detection results from 3D dense LIDAR to planar LIDAR. The resulting 3D object detection algorithm is expected to better deal with the problem of overfitting and be applicable over a wider range of datasets and scenes.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," pp. 6526–6534, 07 2017.

[2] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," 12 2017.

[3] S. Song and J. Xiao, "Deep Sliding Shapes for amodal 3D object detection in RGB-D images," 2016.

[4] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2017.

[5] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," 2019.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] S. Goswami, "Kitti 3d object detection dataset." `https://medium.com/test-ttile/kitti-3d-object-detection-dataset-d78a762b5a4`.

Kalman Filter-Based Real-Time Detection and Localization of Objects on Conveyor Belt

Chenxiao Yu[1], Aditya Agarwal[2] and Maxim Likhachev[2]

***Abstract*— Real-time, fast recognition and continuous tracking of dynamic objects is a significant prerequisite for automation tasks such as conveyor belt manipulation, which requires precise localization. There have been many achievements in dynamic object detection. However, accurately locating moving objects in 3D space is still a challenge due to random errors. To reduce errors, this paper makes improvement on an existing perception system by applying Kalman filter [1] to the localization part. The system takes its input from an RGB-D camera, and publishes the coordinates of objects as they are detected. In the experiment, we had the perception system to detect one object placed on a moving conveyor belt. The accuracy of localization was evaluated by analyzing the errors. Then we compared the performance of the improved system with the original one to verify the effectiveness of our improvements. The results show that the average errors are reduced by 36.27% on x-axis, 16.67% on y-axis, and 35.29% on z-axis.**

I. Introduction

The application of robot in automation manipulation is mainly based on Hand-Eye Coordination [2]. Robots receive visual information and process it through a perception system before transmitting specific data to its controller. According to the data, the controller drives the robot to do corresponding actions. In this process, the correctness of the data is important because it decides whether robots can complete tasks. Therefore, improving the accuracy of data generated from visual information, including object type, position, and orientation, has been a significant goal.

In this paper, we are going to discuss one automation manipulation scenario where data from perception must be highly accurate, conveyor belt manipulation. In this situation, the robot should continuously determine the position of the object on a moving conveyor and plan to grab it. The input visual information is point clouds of the object and conveyor from an RGB-D camera, Figure 1. The perception system then handles the information and publishes the coordinate of the object centroid in robot's base frame. However, the coordinates published are usually inaccurate due to random errors caused during the whole process. Such errors that can be also regarded as noises are unavoidable and will affect the followed planning and grabbing part.

To solve the problem stated above, we apply the real-time Kalman filter to the localization, which is the final part before publishing data. The filter is able to estimate the values that are closer to true ones based on the previous data from localization. The experiment verifies that this method filters out extreme values and greatly reduces the average errors.

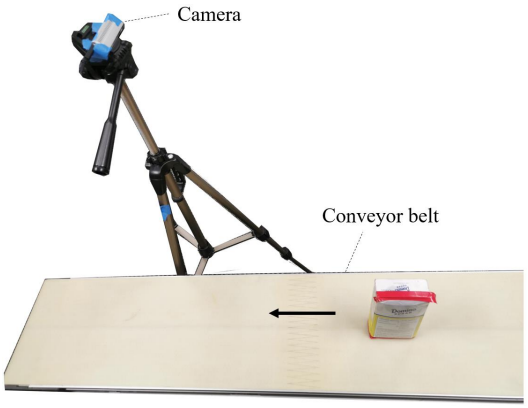


Fig. 1. The point clouds of a cuboid placed on conveyor belt are took from an RGB-D camera.

The rest of the paper is organized as follows: Section II discusses related work. Section III introduces the pipeline of the perception system. Section IV explains the improvement of applying Kalman filter to localization. Section V describes the experiment process. Section VI shows the results and analysis of the experiment.

II. Related work

Localization accuracy is highly required in multiple cases so there have been many methods to reduce the errors. When using a localization method based on landmarks, to solve the data association problem, the nearest neighbor filter has been used to correctly assign the landmarks detected by the sensors to the landmarks present in the map [3]. In intelligent transportation, it took the carrier-to-noise ratio (CNR) of raw pseudorange measurements into consideration for mitigating noises so that it can improve the accuracy of the distance detection [4]. A method named PERCH [5] was introduced for multi-object recognition and localization, which is able to correctly estimate the pose of over 20 objects with translation error under 1 cm and rotation error under 5 degrees.

Except for the methods mentioned above, Kalman filter is also widely used in the intelligent field. In dynamic object tracking, Kalman filter has been used for extracting the estimate of object's position on the table plane [6]. An extended Kalman filter (EKF)-based algorithm has been

[1]Robotics and Artificial Intelligence Laboratory, Chinese University of Hong Kong, Shenzhen , China. This work is done during Robotics Institute Summer Scholar program in the Robotics Institute, Carnegie Mellon University. 116010273@link.cuhk.edu.cn

[2]Robotics Institute, Carnegie Mellon University, USA.

applied to real-time vision-aided inertial navigation [7]. A position estimation scheme for cars based on the integration of global positioning system (GPS) with vehicle sensors also involved extended Kalman filter [8]. In addition, the extended Kalman filter has been employed to identify the speed of an induction motor and rotor flux based on the measured quantities in a vector control [9]. Inspired by the various application of Kalman filter, we consider to integrated it into the existing system to improve localization accuracy.

III. Perception System Structure

The perception system is based on RGB-D sensor data and the base frame of UBTech's Wheeled Walker robot, Figure 2. The overall procedure is that input point cloud data is transformed from camera frame to base frame. The region without conveyor is cut out to avoid interference from other objects. Then conveyor geometry is extracted and the remaining points are clustered. Those clusters are recognized and located to provide coordinates in the base frame. The main parts of the perception system are described in detail below.

A. Frame Transformation

Transforming the point cloud to robot base frame is necessary here, Figure 3. First, the output of the perception system is for robot grabbing so the coordinates should in robot frame. Besides, making the conveyor belt parallel to XY plane can simplify the process of cutting work region and improve recognition performance. The transformed frame is a right-hand coordinate system with the X-axis extending forward from the robot, the Y-axis extending to the robots left, and the Z-axis extending up from the base.

B. Conveyor Surface Extraction

In this step, we take advantages of the functions in PCL segmentation package to filter the conveyor surface. We set the model type as PLANE and using RANSAC [10] method to search for points which belong to the conveyor. The distance threshold, which determines how close a point must be to the plane model in order to be considered an inlier, is set as 0.04m. After plane extraction, the points considered as inliers are removed. This process repeats until the number of remaining points is less than 30% of the original one. Finally, all the points that are not within the object are filtered out.

C. Recognition and Localization

Since the surface of the conveyor has been removed, The remaining points form the point cloud of the object to be detected. They are clustered using kd-tree search method. Once a cluster is formed, we still use RANSAC method to extract the points which are considered inliers of a cylinder. Although what we are going to detect is not always cylinders, this cylinder fitting method has been verified to be able to recognize all the objects with regular shape, including cuboids. Once a cluster is successfully fitted by a cylinder, we can compute the centroid of the cylinder, which is exactly the position of the object.

IV. Kalman Filter-based Localization

The frequency of localization is about 20Hz, which is high enough for the robot to plan to grab. However, there exists randomness in the process of cylinder fitting process, so that the centroid coordinates computed is not always accurate. During the tests, when the object moves towards the camera, the values of x, y, and z axes keeps fluctuating and some particles obviously deviate from normal values. These outliers will cause a bad impact on grabbing planning. In order to optimize the localization performance, a real-time filter is needed to prevent the position value from being interfered by random error. Here we apply Kalman Filter to the localization, which is suitable for linear, discrete and finite-dimensional Systems. The filter will produce estimates of the true coordinate values that are more accurate than those only based on the detection. The algorithm is shown in Algorithm 1.

Algorithm 1 Kalman Filter-Based Localization

Input: State-transition model, A
Observation model, H
Predicted error covariance, P
Covariance of the process noise, Q
Covariance of the observation noise, R
Filtered value at time step k-1, V_{k-1}
Detected value at time step k-1, V_d

Output: Filtered value at time step k, V_k

1: Make a prior estimate $V_{est} = A * V_{k-1}$
2: Calculate prior mean variance $P = A^2 * P + Q$
3: Compute Kalman gain $K_G = P * H/(P * H^2 + R)$
4: Calculate posteriori estimate $V_k = (V_d - V_{est}) * K_G$
5: Update posteriori estimate $P = (1 - K_G * H) * P$
6: **return** V_k

V_d and V_k respectively denote the detected coordinate after localization process and output one after Kalman filter. V_k is given an initial value at the beginning according to measured x,y, and z. The values of A, H, and P are set to certain values based on experience. R is obtained by computing covariance using experiment data, which reflect the noise of the perception system. The value of Q depends on R and the actual test situations. This algorithm is applied to x-value, y-value, and z-value respectively. Each time the coordinate of object centroid is obtained by detection, the values of x,y,z are input to three filters to compute the estimates of true value. Thus, the system is able to publish filtered values with the same frequency as detected ones, which achieves the goal of real-time particle filtering.

V. Experiments

A. Notations and Parameters

In this stage, we are going to test the effectiveness of Kalman filter. The parameter settings of the Kalman filter is shown in Table I. Since the values of x, y, and z are input to three filters respectively, they have different Q, R and initial values. The value of A, H, and P are set as the same

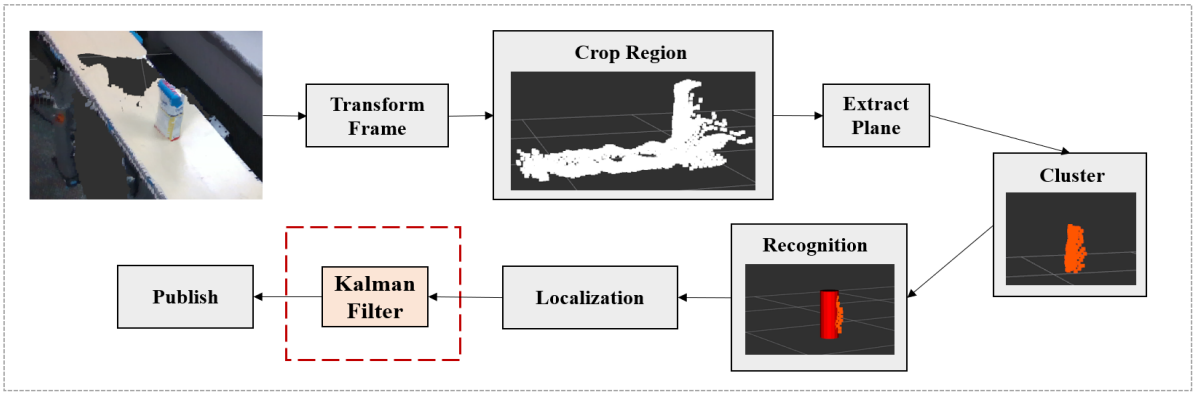
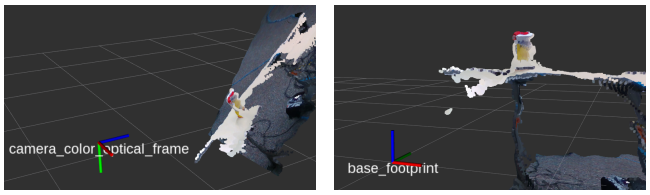


Fig. 2. The pipeline of the perception system. A filter is added after the localization part before it publishes coordinates.



(a) Camera Frame (b) Base Frame

Fig. 3. Frame transformation. (a) shows the point clouds in camera's frame and (b) shows the point clouds in base frame.

TABLE I
EXPERIMENT PARAMETERS

Parameter	Definition	Value
A	State-transition model	1
H	Observation model	1
P	Predicted error covariance	2
Q_x	Covariance of process noise on x axis	0.005
Q_y	Covariance of process noise on y axis	9×10^{-6}
Q_z	Covariance of process noise on z axis	7.6×10^{-7}
R_x	Covariance of observation noise on x axis	0.0284
R_y	Covariance of observation noise on y axis	1.86×10^{-4}
R_z	Covariance of observation noise on z axis	3.61×10^{-5}
V_{0_x}	Value of x coordinate at time step 0	1.4 m
V_{0_y}	Value of y coordinate at time step 0	-0.15m
V_{0_z}	Value of z coordinate at time step 0	0.71m

because they are estimates that has little influence when the filter works.

B. Procedure

We aim to evaluate the accuracy of Kalman filter-based localization and compare it with the unimproved one. Therefore, we focus on the output coordinates of only one object and compute the error between output value and actual value.

In the beginning, a cuboid is placed at a start point on the conveyor where the detected x value is larger than 1.25m. Localization and filter will start working when conveyor moves at a speed of 3.2cm/s. Both the detected coordinates and filtered coordinates will be recorded when detected x value is smaller than 1.25m because the perception system has bad performance when the cuboid is too far away. Through doing linear curve-fitting on detected data, we can get the speed of the cuboid in x-axis direction and y-axis direction, which are -0.0319m/s and -0.003m/s respectively. Since the height of conveyor is fixed, we regard the speed in z-axis direction as 0m/s. The coordinate of the start point is measured using "Publish Point" in RViz, which is (1.245,-0.155, 0.71). Therefore, the actual coordinates during cuboid moving are:

$$\begin{aligned} x &= -0.0319t + 1.245 \\ y &= -0.003t - 0.155 \\ z &= 0.71 \end{aligned} \tag{1}$$

where t(s) denotes the time when detected and filtered coordinates are generated and t is 0 when the cuboid is at start point. During the experiment, we use ROS Time to record the time of each detection, whose frequency is about 20 Hz.In this way, we can simultaneously get detected value, filtered value, and actual value at each time step.

VI. RESULTS AND ANALYSIS

In this section, we will analyze the experiment data to assess the localization performance after applying Kalman filter.

Letting the cuboid move from start point to endpoint during the experiment, we obtained 352 sets of coordinate values, Figure 4. According to the figure, the previous perception system already performs well in x-axis direction localization but not well for y-axis and z-axis. After applying Kalman filter, the localization performance got significant improvement. The curves generated by filter have less fluctuation and deviation compared with detected ones. This means the filter effectively filtered many extreme values and made estimates that closer to actual values.

In order to do a more rigorous analysis of the accuracy of the localization, we compute the error of detected values and filtered values with regard to actual values, Figure 5. Although the errors of filtered values are larger than errors of detected ones in some time periods, the filter does reduce the error by looking at the general trend. In addition, the filter obviously reduced the maximum error value in the whole process. We also compute the average absolute errors under

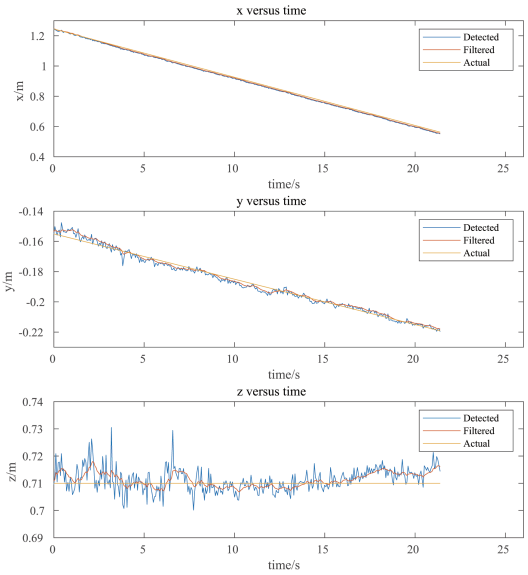


Fig. 4. From top to bottom, x, y, and z values of detected(blue), filtered(red), and actual(yellow) coordinates versus time are plotted.

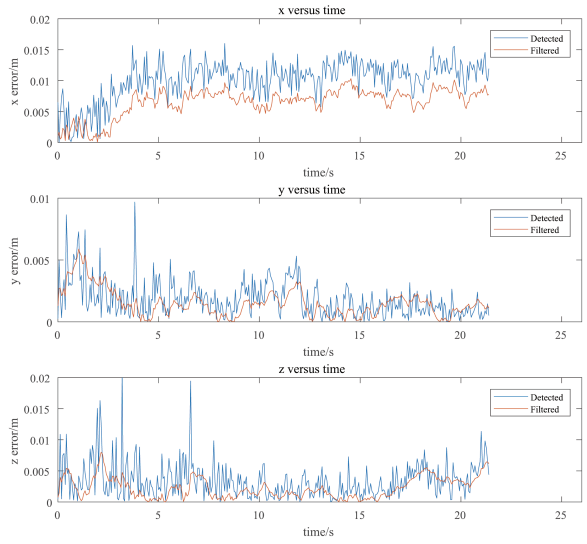


Fig. 5. From top to bottom, errors of x, y, and z values of detected(blue), and filtered(red) coordinates with regard to actual ones versus time are plotted.

these two conditions, Table II. It can be calculated that the Kalman filter reduces the average error by 36.27% on x-axis, 16.67% on y-axis, and 35.29% on z-axis. Therefore, the filter greatly improves the accuracy of localization.

VII. Conclusion

We have improved the localization accuracy of an existing perception system by applying Kalman filter, which aims to recognize the object on a moving conveyor and compute the position of the object. The experimental results show that the modified system is able to locate the object much more accurately. According to the curves of the object position varying with time, it fluctuates less and deviates less. Besides, most of the extreme values are filtered out and the error between published coordinates and actual ones was reduced by 16.67∼36.27%.

This experiment was conducted when conveyor moves at a low speed. In the future, we will verify the system performance when conveyor moves faster and figure out a better solution to keep the accuracy of localization. In addition, the detection of object orientation will be integrated into this system for output more information of the object.

TABLE II
Average Errors

Condition	Error Value (m)
Detected x axis	0.0102
Filtered x axis	0.0065
Detected y axis	0.0018
Filtered y axis	0.0015
Detected z axis	0.0034
Filtered z axis	0.0022

Acknowledgement

I would like to thank Maxim Likhachev and Aditya Agarwal for their mentoring and guidance. Furthermore, I would like to thank Rachel Burcin and John Dolan for their efforts in organizing impressive RISS program. Finally, I would like to thank Robotics and Artificial Intelligence Laboratory, Chinese University of Hong Kong for providing sponsorship and supporting this program.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[2] R. S. Johansson, G. Westling, A. Bäckström, and J. R. Flanagan, "Eye–hand coordination in object manipulation," *Journal of Neuroscience*, vol. 21, no. 17, pp. 6917–6932, 2001.

[3] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[4] K. Liu, H. B. Lim, E. Frazzoli, H. Ji, and V. C. Lee, "Improving positioning accuracy using gps pseudorange measurements for cooperative vehicular localization," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 6, pp. 2544–2556, 2013.

[5] V. Narayanan and M. Likhachev, "Perch: Perception via search for multi-object recognition and localization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5052–5059.

[6] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 816–823.

[7] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[8] S. Rezaei and R. Sengupta, "Kalman filter-based integration of dgps and vehicle sensors for localization," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 6, pp. 1080–1088, 2007.

[9] Y.-R. Kim, S.-K. Sul, and M.-H. Park, "Speed sensorless vector control of induction motor using extended kalman filter," *IEEE Transactions on Industry Applications*, vol. 30, no. 5, pp. 1225–1233, 1994.

[10] 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.

Implementing Face Recognition on a Social Scrabble-Playing Robot

Vicky Zeng[1] and Reid Simmons[2]

***Abstract*-- It has always been a vision in the field of Human-Robot Interaction for autonomous robots to be useful and accepted in our everyday lives. The ability of a social robot to engage in natural interactions and maintain long-term relationships thus becomes a central focus, and this is also the case for Victor, Carnegie Mellon University's (CMU) Scrabble-playing robot. We believe that personalization will improve engagement and help establish intimate, long-term relationships. A functional key needed before personalization then is face recognition, so that Victor can recognize the players he engages with. While state-of-the-art face recognition is continuously improved in private, commercial systems, little research is done on local optimizations of free, open-source face recognition. This paper conducts a preliminary comparison between two open-source face recognition systems - OpenFace and Dlib - and moves on to optimize Dlib, the selected model. While the Dlib model remains unchanged, classifiers and additional code are put in place to translate the model's inputs and outputs such that predicted labels are calculated differently and potentially overridden. Moving forward, successful implementation of face recognition would allow us to record interaction logs between each player and Victor, and the information can then be used to personalize subsequent interactions.**

***Keywords*—Face Recognition, Human-Robot Interaction, Open Source**

I. Introduction

Home is the final frontier, as some roboticists would say [1]. Though many robots are currently designed to perform specific tasks in unmanned environments, it is always a goal to bring robots closer to society, to our daily lives. Social robots are designed to interact routinely with people, and they are useful in tasks where human labor is limited, such as in the industry, entertainment, education, and service sectors.

Victor, Carnegie Mellon University's Scrabble-playing robot, is designed to be a social gamebot. With a playful, sarcastic personality, Victor will readily taunt his opponents – "Is that all you've got?" – and brag about himself while playing a game of Scrabble.

[1] V. Zeng is studying Artificial Intelligence at Carnegie Mellon University, Pittsburgh, PA 15213 (vzz@andrew.cmu.edu)

[2] R. Simmons is a professor and director of the Artificial Intelligence major at Carnegie Mellon University

Victor's torso is topped with a mobile head, displaying his animated face on a video screen. The Scrabble board is displayed on a table-sized touchscreen around which Victor

Fig. 1: Victor playing Scrabble with other players [2].

and three other players sit. While he has no arms, he can electronically move his tiles while people move the virtual tiles with their fingers. Players can converse with Victor using keyboards. Despite his being a gamebot, Victor is not meant to be a scrabble master – he only has an 8600-word vocabulary compared to the official Scrabble dictionary of 178000 words. His personality and chat feature also make him a social robot.

We believe that for Victor to fit into the CMU community, he needs to conform to social conventions and not the other way around. As a result, Victor's interactions with players need to be natural and long-term. Personalizing interactions for individual players would be a huge step in establishing intimate, long-term interactions, as that will allow Victor to interact with each player like a friend. Before that, Victor needs to be able to recognize the players he interacts with. Keeping in mind the importance of naturalness, we look for the most common way a human recognizes another human – through face recognition [3].

Given a limited budget and the expectation that recognition will run continually, we considered two free, open-source face recognition systems – OpenFace and

face_recognition (python wrapper library of Dlib). For simplicity, we will refer to face_recognition as Dlib in this paper. The Labeled Faces in the Wild (LFW) dataset is a standard benchmark in face recognition. On the LFW benchmark, OpenFace achieves a 92.92% accuracy and Dlib achieves 99.38% [4, 5]. Both are numbers comparable to state-of-the-art recognition models and thus good candidates. In this work, a preliminary comparison was done on the two models with small test sets, where OpenFace was eliminated due to low Asian accuracy. We then optimized Dlib by balancing three desired outcomes: 1) maximize accuracy (correctly labeling known people), 2) minimize false positives (wrongly labeling a known person as another person), and 3) maximize unknown detection (correctly labeling an unknown person as 'unknown'). The Dlib model remained unchanged, but data analysis was done to manipulate the model's inputs and outputs such that original decisions were calculated differently and potentially overridden.

We used the k-nearest-neighbors algorithm for the model, performing a weighted majority vote on the k most similar face images. Different dataset size classes also showed different output patterns. Using these patterns, we generated conditions with respect to dataset sizes. When these conditions were met, predicted labels were overridden with 'unknown'. This was useful in reducing false positives and increasing unknown detection while sacrificing a small percentage of accuracy.

The paper is organized as follows: Section II compares OpenFace and Dlib (background and related work), section III explains the methods and evaluations of our optimization on Dlib, section IV shows our final results, and section V covers the next steps of this research.

II. Preliminary comparison of OpenFace and Dlib

Both face recognition systems use convolutional neural networks and go through similar phases. Given an image, face detection is run to find faces in the image. Next, the faces are preprocessed and fed as input to the neural network, which produces a low-dimensional representation (otherwise known as an embedding). This embedding can then be used with classification techniques to eventually output a predicted label.

For both systems, Dlib's face landmark detection is used to detect faces. Preprocessing is also similar, and the final embeddings are both represented in 128D vector spaces, where distances directly correspond to a measure of face similarity [4, 5]. The main difference between OpenFace and Dlib lie in their neural networks' structure, loss function, and training data. Note that the neural networks described below were already trained by their designers: our work only uses the trained models.

OpenFace uses FaceNet's architecture, utilizing a triplet loss function [4]. This triplet selection procedure picks two images of the same person and one image of a different person, and the network is repeatedly trained such that the numbers of the two similar images are closer to each other and further from that of the different image. Two public datasets CASIA-WebFace and FaceScrub are combined to form 500K images, and the designers trained the network with this dataset.

Dlib uses a ResNet network with 29 convolutional layers. It is similar to ResNet-34 published in Deep Residual Learning for Image Recognition [6]. By the designer, it uses "a structured metric loss that tries to project all the identities into non-overlapping balls of radius 0.6. The loss is a type of pair-wise hinge loss that runs over all pairs in a mini-batch and includes hard-negative mining at the mini-batch level" [5]. A notable difference between Dlib and OpenFace is that Dlib's procedure runs over pairs while OpenFace's run over triplets. Dlib's network was trained from a dataset of about three million images (7485 identities) that was formed from FaceScrub, VGG, and the designer's collection of images from the internet.

We used a small self-collected dataset of 12 identities, five to eight images per identity, to test the two models. The dataset contained a mix of races – White, Black, Asian, and Multiracial. OpenFace performed significantly worse on the Asian faces, averaging an accuracy of 40% (2/5 faces) compared to Dlib's 80%. We achieved similar results with bigger datasets of 30 Asian faces. This was a problem, as nearly 30% of CMU's student body was Asian in 2018 [7]. Upon further inspection, we noticed that the dataset used to train OpenFace's neural network lacked Asian identities, and other users have similarly reported low Asian accuracy [8].

The designer of OpenFace made it possible to retrain the neural network with our own dataset. To have an idea on what a good size for a training set would be, we referred to Facebook's DeepFace network, which was trained on a private dataset of four million images [4]. Unfortunately, training was too computationally and memory expensive that it would take up to a month for a dataset that's half the size of Facebook's (at two million images). That made training infeasible with the time

constraint, so Dlib (face_recognition) was chosen over OpenFace.

III. Methods and Evaluation

Recall that the goal is to add optimizations that improve overall performance without having to modify the Dlib model itself. Performance is improved by maximizing accuracy, minimizing false positives and maximizing unknown detection. This was a continuous process, as the outcomes gradually improved for each new optimization introduced. This section will follow a similar flow, where we explain each optimization and the improvement brought about by that addition. For the final results (achieved through the combination of all optimizations), refer to the next section, section IV.

A. Datasets

All the datasets we used to test Dlib were subsets of two datasets – MS-Celeb-1M-vlc and Asian-Celeb. MS-Celeb-1M-vlc is a cleaned subset of MS-Celeb-1M, a large image dataset made available to the public by Microsoft. It contains the top 100K celebrities, who are mostly western [9]. The cleaned subset MS-Celeb-1M-vlc we used contains 86K celebrities with close to four million aligned images. The other dataset we looked at was Asian-Celeb from glint [10]. As the name suggests, it contains 94K Asian celebrities with around three million aligned images. This is the largest, free Asian dataset available online with a good portion of identities having over 25 images each, and there contains little to no overlap between these 2 datasets. For each of my dataset, half of the identities came from MS-Celeb-1M-vlc while the other half came from Asian-Celeb, to reflect CMU's race demographics.

B. First Optimization: Recalculating predicted label to increase accuracy

For simplicity, our datasets only had 1 face per image. We define a prediction to be accurate when the model correctly labels the known face in the image. Accuracy is then calculated out of all images with known faces, dividing that by the number of correct labelings of known faces.

Datasets of different sizes were used to validate accuracy consistency within and across datasets. They were also used in a later optimization to find patterns. The sizes were 30, 200, 400 and 600 identities respectively.

Existing limitation

The method provided by the model only took one labeled image per identity label. However, many factors such as light illumination and facial expression variations could affect the similarity between two faces, making one image per label potentially problematic [3]. When given an image to predict, it found the best image match and output the corresponding label as the prediction. With this technique, we were limited to only returning the top image match, since all other matches would be of different labels and thus add no value.

Baseline results (before any optimization)

The default method produced 83.5% accuracy and 11.4% false positives (the remaining 4.6% were incorrectly labeled as 'unknown'). However, there was a high standard deviation of 2-3% across all numbers. Since training images were randomly chosen and there could only be one training image per label, the quality of training images varied each run, affecting the numbers.

Solution

This problem was solved by applying classifiers to the model – instead of having one image per label, multiple images could be associated with the same label. With this, we could meaningfully look at subsequent image matches after the best match, as the images after could be of the same label or not. This allowed us to observe patterns through data analysis and testing.

We define the following terms for clarity. A label is the identity of a person (i.e. Vicky), an image is an instance of a label (there can be multiple images of Vicky in the dataset), while the distance is the distance between an unlabeled image and labeled image (the smaller the distance, the more similar the two images are). This distance is a numerical value calculated by the model.

Given a dataset, we randomly split it into training and testing data with a 7:3 ratio. Training data were used to train the classifier, so each training image was labeled. Testing data were unlabeled where the Dlib model predicted the labels. Using a dataset of 200 identities, we compared a Support Vector Machine classifier with a K-Nearest Neighbors classifier. An SVM classifier creates a hyperplane which separates images into classes [11]. In k-NN, the image is classified by a plurality vote of its k nearest neighbors and assigned to the label most common among these neighbors [12]. Refer to Figure 2 for a visual representation of the classifier.

The SVM classifier produced an accuracy of 91% while the k-NN classifier achieved 93% with k=3.

Now using the k-NN classifier, we explored three factors: the number of training images per label (5,10,20), the k value (3,5, $\sqrt{\#identities}$, $\sqrt{\#dataset\ size}$), and the weights to the

respective neighbors (1, $\frac{1}{distance}$, $\frac{1}{distance^2}$) . When testing a factor, other factors were kept at the optimal values found previously. If the factor was yet to be optimized, the default value was chosen.

Results after the first optimization

The best combination showed a k-NN classifier with 10 training images per label, *k*=5, and a weighted vote of $\frac{1}{distance^2}$ for each neighbor. Accuracy increased by 11.3% to 94.8%. The remaining 5.2% were false positives.

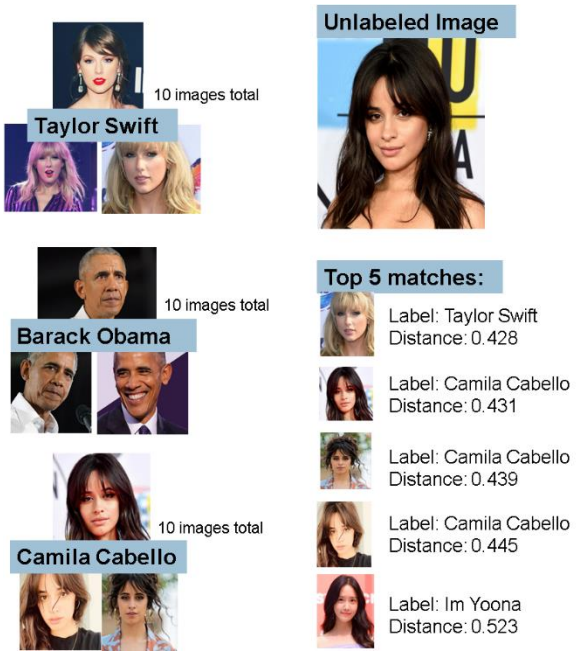


Fig. 2: Sample run with an unlabeled image. Images from [9].

C. Second Optimization: Overriding predictions to reduce false positives and increase unknown detection

From the optimization in part B above, we got 94.8% accuracy and 5.2% false positives on the dataset of 200 labels – 2000 training images and 600 test images. 5.2% false positives meant that 5.2% of the known images were instead labeled as another label. That was undesirable in a real-world application. Imagine recognizing a friend as another friend. Instead, it would be ideal that uncertain faces be labeled as 'unknown'. Similarly, we want unknown faces to be labeled as 'unknown'.

Existing limitation

The intuitive solution was to lower the accepted threshold on distance, where faces were labeled 'unknown' if they were above this threshold (the greater the distance, the more different two images were). The idea followed because distance was supposed to be a measure of similarity. Unfortunately, at this level they no longer were good measures of uncertainty: lowering the threshold compromised a significant percentage of accuracy, reducing up to 10% accuracy for reducing 1% false positives. This suggested that the model's measure of similarity was inaccurate at this level.

The model was also bad at detecting unknown faces. Two unlabeled datasets of 20 labels (60 images) and 150 labels (300 images) were used to test unknown detection. At this stage, unknown detection was 64.4%.

Method

To distinguish true positives from false positives and correctly identify unknown faces, we needed to find other patterns indictable of uncertainty. Conditions based on such patterns could then be written to detect and override the model's predicted labels to 'unknown'. Large amounts of testing and data crunching were done on the top nearest matches to find patterns.

The challenge was three-fold as there were two sources of information from the top 5 image matches: the distance values and the identity labels. This meant that we could find patterns from the distance, the labels or a combination of the two. Given the vast number of possible combinations to check, going through them systematically would be inefficient. We approached the problem from two directions. The first approach split the false positives from the true positives. Then, the same data collection was conducted on both pools, looking at the distances and labels separately. When any noticeable differences occurred, related combinations of distances and labels would then be looked at. The second direction started by analyzing the patterns in false positives. As the pool of false positives was significantly smaller, more pattern data could be collected at a faster rate. When meaningful patterns appeared, they would be tested against the true positives to assert viability and tweak condition thresholds. This was repeated for unknown detection, with the two pools as unknown faces and known faces.

Solution

The conditions written based on the patterns found could be described as three types. Some conditions were local – they were based on patterns unique to that dataset size range. Other conditions were an in-between – the pattern was repeated, but the exact threshold for the condition to override predictions to 'unknown' was different. The remaining conditions were universal – they were based on patterns repeated across all dataset

sizes.

In consideration of the complexity and number of conditions, this paper will just highlight the general trends of different condition types and some noticeable differences between local conditions of different dataset sizes. This is our second and last optimization, and results will be displayed in the next section.

Local conditions

Within the top five image matches, recall that each match was associated with a label and a distance value. The patterns we found could be categorized into three groups: patterns on unique labels, the threshold of distances, and the distance difference between matches.

1) *Unique Labels*

One pattern we actively looked at was the number and order of unique labels within the five matches. For example, if the five matches contained three images of Vicky and two images of Angela, there would be two unique labels as the number. With the same example, we could list the five matches in increasing distance value: Angela_image_1, Angela_image_2, Vicky_image_1, Vicky_image_2, Vicky_image_3. Then we would observe that although there were more images of Vicky, Angela's images were more similar to the unknown image, and this was an order of unique labels we would be concerned about.

Smaller dataset sizes of 100 and below achieved better results focusing conditions around unique labels. As there were fewer labels stored in the dataset, it was less likely for a known face to be matched with multiple other labels. This was not true for dataset sizes of 100 and above. As the number of labels increased, a bigger portion of correct predictions was also matched to multiple different labels, so revolving around unique labels was not the best approach to separating false positives and unknown images from true positives. Dataset sizes in the 100-400 range still had patterns partially related to unique labels, but dataset sizes above 400 did not.

2) *Threshold on the distance value*

Threshold referred to a cutoff on the distance value. Recall that each matched image had a distance value, representing the distance between that image and the unlabeled image. The greater the distance, the less similar the two images should be. As mentioned previously, dataset sizes in the 100-400 range relied partially on the number of unique labels but their conditions were less strict than the smaller datasets. That was because a good portion of correct predictions also had a big number of unique labels, just like false positives. For these datasets, their sizes were manageable enough for the Dlib model such that, after pattern-matching on the unique labels, playing around with thresholds could efficiently distinguish the false positives and unknowns. This could not hold for dataset sizes above 400. Due to a large number of labels, it was common for unlabeled images to have close distances with at least some of the labeled images.

3) *Distance difference between matches*

For dataset sizes above 400, the large number of labels and images meant that many correct predictions would be matched with multiple labels, and many unknown images also had close distance values to some labeled images. The tools directly provided by the model was thus insufficient. In this case, we found more success calculating distance differences between matches – in other words, comparing how similar each matched label was to the unlabeled image. As an example, we could have the following five image matches in ascending distance value: Vicky_0.40, Angela_0.41, Angela_0.42, Vicky_0.42, Vicky_0.43. Then, we would try several different calculations. We could take the average of each label (i.e. Vicky: 0.4+0.42+0.42 / 3 = 0.413) and extract that difference. We could first categorize the examples into different groups (i.e. one group contained all examples where the first label was the majority among top five matches but a minority among the top 3 matches), then take the average of the labels. After computing the numbers, we would find the difference between each label pair. If the difference was small, then it suggested the labels had the same level of similarity with the unlabeled image. The prediction would then be overridden with 'unknown'.

In-between conditions

The logic for in-between conditions followed by that of the local conditions. From the local conditions, we got that there were three main factors: unique labels, the thresholds, and the distance differences between matches. In-between conditions had a good combination of these three factors, but the threshold for each dataset size range was slightly different to match their situations. This was the case because the greater the number of labels, the greater the probability of an unknown image having close distance values with a labeled image. Since we were sacrificing accuracy for false positives and unknown detection, we needed to take advantage of this probability and maximize the trade-off (preserve as many correct predictions as we could while keeping false positives and unknown images above the threshold). Therefore, the threshold value changed depending on the dataset size.

Universal conditions

Although we have described the general trends for each dataset size range above, some patterns existed across. A sample condition was as follows: If the top 3 matches were all unique labels, override the prediction to 'unknown' if the difference between the first two matches was lesser than 0.08. On a high level, universal conditions indicated an extent of reliability of the face recognition model itself, as it did not rely on local observations to fine-tune patterns.

IV. Final Results

The table below displays the results before and after all optimization. Note that the unknown column is different from the unknown detection column. The unknown column refers to all known faces that are predicted as 'unknown'. This number is a combination of reduced accuracy and reduced false positives. The unknown detection column refers to all unknown faces that are correctly detected as 'unknown'. Unknown faces are from a separate dataset and are not included in the calculations you see in the accuracy, false positives and the unknown columns.

TABLE I: Results before and after optimization. The numbers are averaged from five runs of each dataset, and the numbers in parentheses represent their standard deviations.

Dataset size	Accuracy (%)	False Positive (%)	Unknown (%)	Unknown Detection (
200 (BO*)	**83.5 (3.43)**	**11.4 (2.25)**	**4.6 (1.21)**	**44.4 (2.5**
200 (WC)**	**94.8 (0.75)**	**5.2 (0.62)**	**0.2 (0.39)**	**64.4 (1.6**
20	95.8 (1.90)	1.2 (1.15)	2.4 (1.46)	93.7 (1.8
200	**89.1 (0.63)**	**0.8 (0.26)**	**10.1 (0.39)**	**89.4 (1.5**
400	88.3 (0.85)	1.0 (0.19)	10.6 (0.81)	88.2 (1.1
600	87.9 (0.71)	1.2 (0.15)	10.9 (0.8)	86.7 (1.2

*BO: Before optimization **WC: With classifier optimization only

Compared to before optimization, there was an improved performance across all ends – accuracy increased by 5.6%, false positives reduced by 10.6%, and unknown detection increased by 45.2%.

The high standard deviation before optimization was explained earlier in the paper. As only one training image per label was taken, a random split of training and testing data caused the quality of training images to vary each run, creating large standard deviation.

After optimization, most datasets had similar standard deviation except for the dataset of size 20. Due to the small dataset, the testing set was also small, so every wrong labeling took up a bigger percentage compared to wrong labeling in dataset sizes of 200 and above.

Notice that accuracy was highest with only a classifier, standing at 94.8% compared to the final 89.1%. This decrease in accuracy was the trade-off we made to reduce false positives from 5.2% (with classifier only) to 0.8% (after all optimization). It was a worthwhile trade-off.

V. Future Direction

Going forward, we look to integrate face recognition into Victor's current system. Physical responses from Victor can then be considered, such as greeting and looking in the direction of the person.

Database structures are currently set up to support storage of interaction logs and game data. With face recognition, statistics from recognized players can then be stored across games and be used to personalize subsequent interactions.

The general approach is to observe patterns from a player's playstyle, chat history, visit frequency and respond to changes in those patterns. Specifically, we are looking into four areas: game, time, interaction and external.

For game-focused patterns, we can look at the player's skill level (score), his response times for each turn, and the types of turns he makes (double, triple, passes). For time-focused patterns, we can look at total game duration, visit frequency, and days and times of visits. For interaction, sentiment analysis can be done to change Victor's level of snarkiness. Other ideas include having simple conversations on past, current or upcoming events. Lastly, for external areas, we are looking to have Victor reach out to players through online platforms, such as inviting them to play during de-stress weeks over the semester. There remains a lot of potential for exploration.

VI. Acknowledgment

I would like to thank Dr. Reid Simmons for his guidance and support on this project, and Jenny Han for her support and accompaniment. I would also like to thank Rachel Burcin, Dr. John Dolan, and Mikayla for organizing RISS and making this all possible.

VII. References

[1] K. Weir, "The dawn of social robots", *American Psychological Association*, 2018. [Online]. Available:

https://www.apa.org/monitor/2018/01/cover-social-robots. [Accessed: 03- Aug- 2019].

[2] M. Lynch, "This CMU-Built, Trash-Talking Robot Wants To Beat You At Scrabble", *WESA*, 2014. [Online]. Available: https://www.wesa.fm/post/cmu-built-trash-talking-robot-wants-beat-you-scrabble#stream/0. [Accessed: 03- Aug- 2019].

[3] M. Correa, J. Ruiz-del-Solar and F. Bernuy, "Face Recognition for Human-Robot Interaction Applications: A Comparative Study", in *12th annual RoboCup International Symposium*, Suzhou, China, 2018, pp. 473-484.

[4] B. Amos, B. Ludwiczuk and M. Satyanarayanan, *OpenFace: A general-purpose face recognition library with mobile applications*. Pittsburgh, PA, 2016.

[5] D. King, "High Quality Face Recognition with Deep Metric Learning", *Dlib*, 2017. [Online]. Available: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html. [Accessed: 08- Jun- 2019].

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778.

[7] "Student Enrollment", *CMU*, 2018. [Online]. Available: https://www.cmu.edu/ira/Enrollment/pdf/fall-2018-pdfs/university-facts-2018-student-enrollment-by-citizenship-race-sex.pdf. [Accessed: 05- Aug- 2019].

[8] A. Fydanaki and Z. Geradts, "Evaluating OpenFace: an open-source automatic facial comparison algorithm for forensics", *Forensic Sciences Research*, vol. 3, no. 3, pp. 202-209, 2018. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6201796/. [Accessed 28 June 2019].

[9] L. Zhang and J. Gao, "MS-Celeb-1M: Challenge of Recognizing One Million Celebrities in the Real World - Microsoft Research", *Microsoft Research*, 2016. [Online]. Available: https://www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/. [Accessed: 18- Jun- 2019].

[10] "Face Feature Test/Trillion Pairs", *deepglint*, 2017. [Online]. Available: http://trillionpairs.deepglint.com/overview. [Accessed: 18- Jun- 2019].

[11] R. Pupale, "Support Vector Machines(SVM) — An Overview", *Medium*, 2018. [Online]. Available: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989. [Accessed: 05- Jul- 2019].

[12] N. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression", *The American Statistician*, vol. 46, no. 3, pp. 175-185, 1992. Available: https://www.tandfonline.com/doi/citedby/10.1080/00031305.1992.10475879. [Accessed 3 August 2019].

Tactile only Active Sensing using Reinforcement Learning

Ziwen Zhuang, Jianren Wang, David Held

Abstract— **Human has the ability to identify and pick up objects without seeing it, such as grabbing a pen from the school bag instead of cellphone, book, or wallet. These functionality requires no visual input, where there are cases when visual input is not available for robots. Currently, researchers adopting cross-modal network to perform classification tasks [1], [2], or using fixed touching policy to distinguish primitive objects [3]. And in terms of touching task, current method mainly adopted re-grasping policy [4], which start by pinch the object with high-dimension tactile sensor, and perform grasping quality evaluation to determine whether to pick up the object based on current grasping pose. This work proposed a series of neural networks that can actively touch the objects. And we examine the availability of using depth spherical projection as geometric feature which enables the object classification. Also, using reinforcement learning to train the active touching policy improves the classification prediction accuracy, which shows the concept that active touching with only tactile sensing information is viable in objection classification.**

I. Introduction

Human beings can recognize and grasp object based on several touch information from hand, and perform grasp from different angle with one shot. Considering picking a cup with rings among a groups of pen, book and other accessories, human was able to recognize which one is the cup and pick it up by holing the ring.

However, this ability is not well studied in robotics. The majority of grasping policy is using a two-fingered gripper to pick the object from the top [1], [4]. Some works using multi-modal, which combines visual and tactile as two inputs, to perform instance reorganization [1], re-grasping [4] and object manipulation [5], [6]. The input tactile sensing data are usually high dimensional which makes the sensor data compact enough and quite informative. These training method are mostly non-end-to-end predictors that can be trained by supervised learning with hand-coded loss function, where all touching and sensing policy are pre-defined.

As the target of this work, we proposed a grasp in the dark problem. The robot has no visual input, but are asked to recognize an object whose rough location is already known. The input demand could be the image of the object, a snippet of text, or simply an object class label. The robot should control the dexterous hand with tactile sensors and actively touch the object from different direction. The proposed method would contains three modules: active touching module, feature extraction module and discriminator module. Active touching modules will output the touching direction in high level, which will be executed by a low-level controller. The touch sensors will then returns the tactile data to the feature extraction module. The feature extraction module will be adopting spherical projection and build a sparse depth image as feature output. And the discriminator takes in the feature map and make predictions.

This work proposed a depth feature map that contains the information of object shape information. And the extracted information is indeed helping to distinguish the class of the object. And this work shows the benefit of using active touching policy trained using reinforcement learning, which improves the feature extracted by spherical projection and improves the prediction quality of distinguish network.

II. Related Works

a) Tactile Sensing: Tactile sensors are new method coming into robotics and they are still not well studied. It mainly serve as the combinations with the visual input, which are trained in the multi-modal network and perform re-grasping [4], assessing grasping quality [7], one-shot grasping task [2] and objection classification [1]. Also, tactile sensing network can be tested independently while trained by a pre-trained vision based feature extraction network, that helps guiding the tactile feature network to generate proper feature representation and perform objection recognition task [8]. However, there are few works focus on active sensing using tactile information only, but more of using it as a method of assessing the quality of grasping and aid the robot to perform grasping and re-grasp [9].

b) Active Learning: The main hypothesis in active learning is that the learning algorithm is able to learn from the history of data and determine a policy that skews the data sample distribution, which outperforms the traditional data collecting algorithm. Active vision input is frequently studied starting from 2000, recognizing objects based on the history images [10] and adopt end-to-end policy learning for active vision [11]. Since training the classification network requires backpropagation which is unable to be provided based on the nature of environment noise, reinforcement learning is a proper solution to help bridge the gap between active sensing policy and the classification network. Previous works mostly studied on visual input and finish the reconstruction task using recurrent neural networks [12]. But there are still limited work study around the active tactile sensing policy and recognition.

III. Method

In this section, we described our proposed active touching structure which forms a circle interacting with the environment. Then, in the following subsections, we described in details how we setup feature extraction and feed the feature to classification network as well as using the distinguish accuracy to train the active sensing policy.

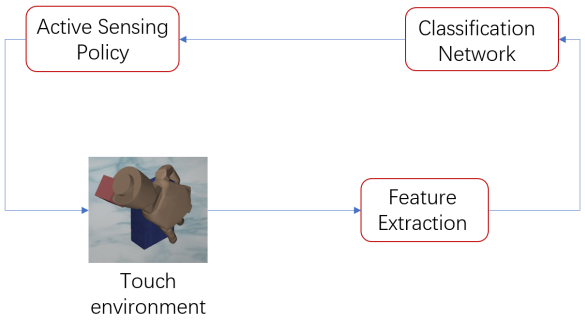


Fig. 1. Proposed active sensing problem structure

A. Feature Extraction

Spherical Projection: In order to feed the classification network with more reasonable feature, where the feature can be accumulated as the new touches comes in and be more informative, we adopted a spherical projection. By acquiring each joints position of the dexterous hand, these 3D coordinates can be translated to spherical coordinates as figure 3. By taking the ϕ and θ, we will get the pixel coordinates in the spherical map within the $0 \sim 2\pi$ and $0 \sim \pi$ scale[1]. And the depth can be extracted directly from the r value of the coordinates. The depth image would show reveal the geometric awareness of the touching object. From that perspective, each new random touch would generate more contact points. And the new contact points are more likely to be projected onto the pixels that there are no projections before so that new depth information will be added to the spherical depth map.[2]

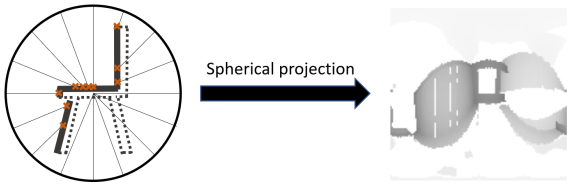


Fig. 2. Spherical Projection from contact points to a predefined sphere

B. Classification Network

Classification from image has already being well studied and generated various of application in industry. Different types of classification network are frequently used, such as AlexNet [13], VGG [14] and ResNet [15]. We are using ResNet 18 to perform the classification task. The classification network c_θ would output a vector based on the input tactile feature map $f^{(m)}$, which gives the score of each possible object. And we use cross entropy loss to compute the network classification error between the ground truth label $f^{(l)}$, which then can be used to back-propagates and tune the network parameters. And the algorithm is tend to compute the following optimization problem:

$$\underset{\theta}{\text{minimize}} - \sum_i f_i^{(l)} \log c_\theta(f_i^{(m)})$$

C. Active Sensing

We proposed an active sensing policy that would take in the feature map and outputs the hand closing gesture as shown in figure 3. The high-level command will be executed by a controller described in the next section. The dexterous hand will make contact with the given objects and return the 3D locations of each contact points as the sensor information. And the sensor information goes to a feature extraction layer which was described in section III-A. The output feature will then feed into a classification network, whose accuracy can be used as a reward for the training of active sensing policy.

D. High to Low Level Touch control

We use the South Hampton Hand Assessment Procedure (SHAP) test suite built by Light et. al [16] and using pre-modeled mujoco file from [17], which gives the testing freedom we need.

The high level control command contains 1 for the hand rotation and 2 information related to the spherical coordinates, and we can denote the action input as a 3D vector $\boldsymbol{a}$. As shown in figure 3, we define $\boldsymbol{a} = (\gamma, \theta, w)$. The reason why we didn't take in r as one dimension of input is that, we are assuming the object size is not given to the robot; there is not point to stop the dexterous hand at a given point; we can keep moving the hand close to the center until the hand's trajectory is being blocked by the object. So, we need only 2 Dof which gives us the closing direction of the hand.

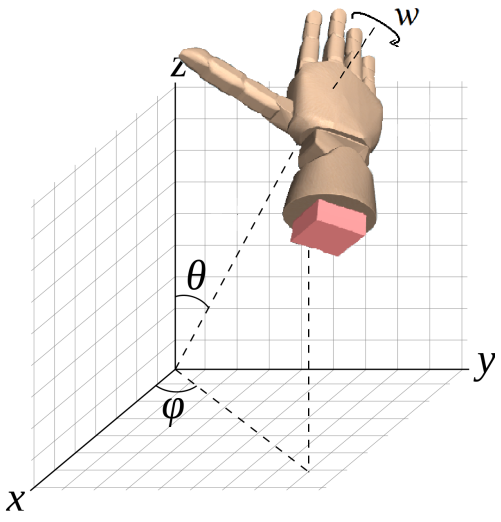


Fig. 3. Spherical Coordinates

Also, the high-level action command gives us the expected direction the agent wants the hand to be. It can be transformed to yaw-pitch-yaw Euler angle by equation (1).

[1]In terms of the image indices, a little shift won't affect the image information

[2]Considering the convolution layer of neural network has the ability of doing signal filtering, the maximum depth setting won't matter as long as all the depth measurement is within the range.

$$\boldsymbol{r} = \begin{bmatrix} \text{yaw1} \\ \text{pitch} \\ \text{yaw2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w \\ \theta \\ \gamma \end{bmatrix} \tag{1}$$

Aside from calculating the hand rotation, the high-level action command also tells us the hand closing direction. So, we need a formulation to keep calculating the hand 3D location and change it frame by frame to move the dexterous hand close to the target object. Considering the control interface only allow us to move the hand by its wrist using motion capture, we need a kinematic tree to calculated the actual low-level command to move the wrist.

Before that, we will calculate the palm pose in the world frame. From the hand rotation, we can get is quaternion (w, x, y, z) by equations (2)

$$\begin{aligned} w &= \cos\frac{r_3}{2} \times \cos\frac{r_2}{2} \times \cos\frac{r_1}{2} + \sin\frac{r_3}{2} \times \sin\frac{r_2}{2} \times \sin\frac{r_1}{2} \\ x &= \sin\frac{r_3}{2} \times \cos\frac{r_2}{2} \times \cos\frac{r_1}{2} - \cos\frac{r_3}{2} \times \sin\frac{r_2}{2} \times \sin\frac{r_1}{2} \\ y &= \cos\frac{r_3}{2} \times \sin\frac{r_2}{2} \times \cos\frac{r_1}{2} + \sin\frac{r_3}{2} \times \cos\frac{r_2}{2} \times \sin\frac{r_1}{2} \\ z &= \cos\frac{r_3}{2} \times \cos\frac{r_2}{2} \times \sin\frac{r_1}{2} - \sin\frac{r_3}{2} \times \sin\frac{r_2}{2} \times \cos\frac{r_1}{2} \end{aligned} \tag{2}$$

And, since quaternion and rotation matrix serves the same functionality, they can be transformed simply by a bi-section function. Transforming quaternion to rotation matrix based on hand frame, we can adopt equation

$$\boldsymbol{R} = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-yw) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix} \tag{3}$$

Thus, we can calculate the low-level command input to the hand control. And the sequence of putting the dexterous hand close to the target object is described in algorithm III-D

Algorithm 1 High level to low level control

procedure HAND CONTROL($\boldsymbol{p}$,$\boldsymbol{m}$,h)
  calculate the expected spherical coordinates $\boldsymbol{s}$ using equation (1) and its quaternion $\boldsymbol{q}$
  calculate the rotation matrix $\boldsymbol{R}$ in hand frame using equation (3)
  initial hand position $\boldsymbol{p}_0 \leftarrow \boldsymbol{R}[0, 0, h]^\top$
  calculate the wrist (mocap) position $\boldsymbol{w}_0 \leftarrow \boldsymbol{p} + \boldsymbol{R}m$
  move the hand to the top of $\boldsymbol{p}_0$ with rotation $\boldsymbol{q}$
  move the hand to the position $\boldsymbol{p}_0$
  move gradually close to the object until setting the palm location to $\mathbf{0}$
  close all the fingers until no hand movement is detected[3]
  record and return all joints relative position in world frame

E. Pretrain data collecting

In order to simplify the action space, we provide the robot with 3 dimension of controlling since there are already well designed algorithms to control robot joints in order to reach certain pose. The control action consist of the spherical coordinates[4] and one controlling the hand yaw rotation on the sphere. Then, the hand will get close to the given rough center and close the hand until it cannot move closer because of blocked by the object. The joints position can be detected using predefined sensors, which serve as a feed back to the feature extraction module. Considering the 3 degrees of freedom (DoF) action space is not a infinite space, we can sample the action directly from a uniform distribution.

We have also tested the method where there are 6 DoF for the active sensing policy to output. The action could consist of a 3D location related to the given object center and a 3D Euler angle that specifies the dexterous hand pose. This gives active sensing policy more freedom to decide how to utilize the characteristics of the dexterous hand. Same as the 3 DoF action space, we random sample the from a uniform distribution. However, as shown in figure 5, this method reveals a serious problem that failed in accuracy for both of the classification method, because the hand are highly likely to touch the object using the back of the palm or the root of the wrist where fingers are not able to touch the object before it close to its own limit.

F. Training Touch Agent

As described in figure 1, an active sensing policy is designed to facilitate touching gesture and improve the classification network accuracy. We put the active sensing policy in the environment with the extracted feature map as observation. We adopt Proximal Policy Optimization (PPO) to train the policy. We designed the reward as negative cross entropy loss output by the classification network, which makes a dense reward. This helps avoid the sparse reward problem where the classification network never make correct judgment on the produced depth feature image.

G. Alternatively Training Touching Policy and Classification Network

We proposed an alternating training method to make the active touching policy and classification network cooperate better. Since it is difficult to back-propagate from throughout the entire pipeline, we train the policy network and the classification network separately. We pre-train the classification network with feature image collected by random touch. Then, we train the active touch policy with fixed classification network. It would be reasonable to train the classification network later when the active touch policy converges to some extent. classification network training will be based on new feature data collected while the active touching policy is training. The training procedure is described in algorithm III-G.

[4]Spherical coordinates has 2 degrees of freedom (DoF).

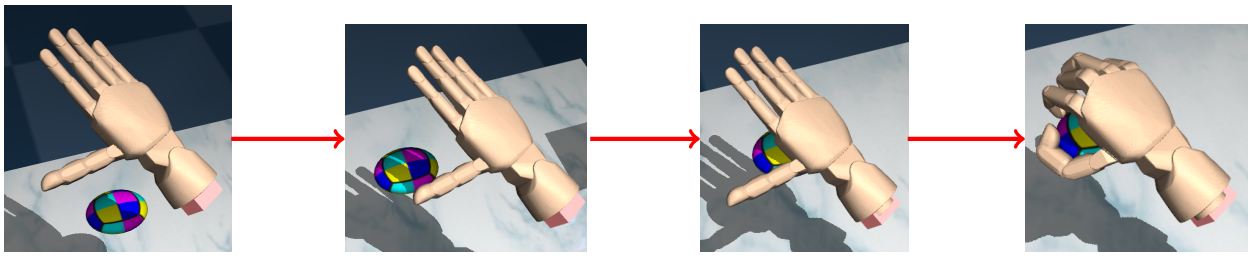


Fig. 4. low-level control of hand touching the object

Fig. 5. 6DoF action space: most of the touches does not make the touch to the object within its palm

Algorithm 2 Alternative training classification network and active touching policy

procedure ALTERNATIVELY TRAINING
  Collect feature image from random touching policy
  Train classification network using feature map
  for $i = 1, \cdots$ **do**
    Collect feature image using active touching policy
    Count the accuracy and loss of classification net
    Optimize active touching policy using PPO
    if $i \bmod t == 1$ **then**
      Train classify net using latest feature image

IV. EXPERIMENT

1) Experiment setup: To build up the test environment, we set up two sets of objects: 5 objects with primitive shapes (displayed in figure 6), and 5 bottles with different shape in details (displayed in figure 7).

Every time we accept a high-level command from the random sample, we move the hand to the top of a chosen object with given pose in case of block during the hand moving trajectory. Then, the hand controlled by motion capture, will go down to the given touch direction. And get close to the object as well as close the fingers, as figure 4. We collect entirely 1000 training touch sequence (or 1000 depth image accumulated by the sequence of touch), where each touch sequence contains 5 finger joints data from a complete high level touch.

Fig. 6. 5 primitive shaped objects

Fig. 7. 10 bottles with similar shape

A. Experiment comparison

First, we compare the availability of the proposed feature map. A simple baseline could be established using Long Short-Term Memory network. Two comparisons are described below:

1) Using hand coded depth feature map extracted from given times of hand touching onto the object. And feed the feature map to the ResNet 18, using cross entropy loss and ADAM [18] as optimizer.
2) Using Long Short Term Memory network, we feed it a sequence of finger joins information and let it learns the feature representation itself, which is the output of hidden state once finish feeding the touch sequence. Then, trained the classification neural network as a fully connected layer, optimizing using ADAM [18] with the target of cross entropy loss.

Then, we test the improvement using what we have proposed. We adopt active sensing policy trained using PPO. And test the classification network using new feature map collect from trained active sensing policy. In order to do the ablation test, we also test the accuracy improvement with only trained the active sensing policy without alternatively tune the parameters of classification network.

V. RESULTS AND DISCOVERY

We tested both using LSTM as feature extraction and hand coded feature extraction method on 5 primitive objects and 5 similar shaped bottles. Both algorithm in both data sets converge well as displayed in figure V

But in comparison with the two algorithm, LSTM takes much more training epoch to reach compatible accuracy than ResNet with hand-coded feature extraction layer. Thus, the proposed spherical feature map is reasonable to provide information and facilitate the training process. However, during the test time, overfit problem is exposed when testing both the LSTM and the ResNet classification network.

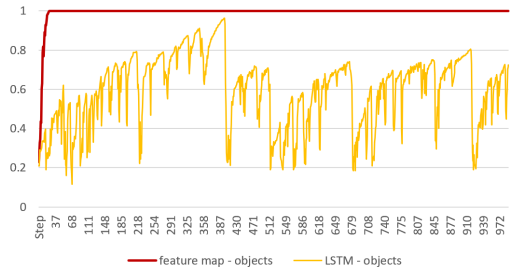


Fig. 8. Accuracy learning curve in 5 primitive objects

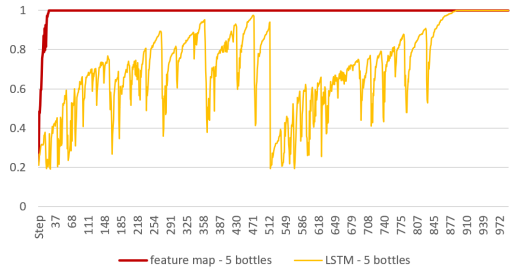


Fig. 9. Accuracy learning curve in 5 similar shaped bottles

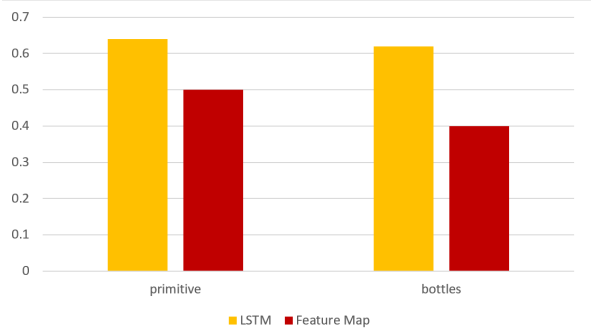


Fig. 10. Test data accuracy in 4 different combination

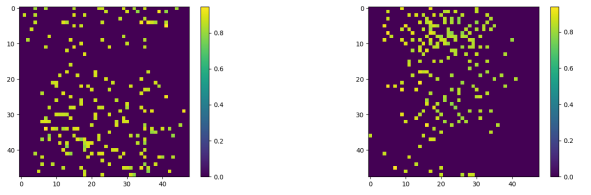


Fig. 11. examples of extracted feature map

As figure 11 shows, the purple part are all non-detected direction, where pixels are initialized with zero. So the depth image matrix might be too sparse so that where the non-zero pixels are positioned, which should be considered as noise, becomes the main factor that influence the classification network output.

By testing the benefit we derived from training active touching policy, we compared the accuracy improvement with and without alternative training the classification network. As figure 12 showed, using trained active touch policy did improve the prediction accuracy of the classification network. But alternative training is need or there will be no accuracy improvement.

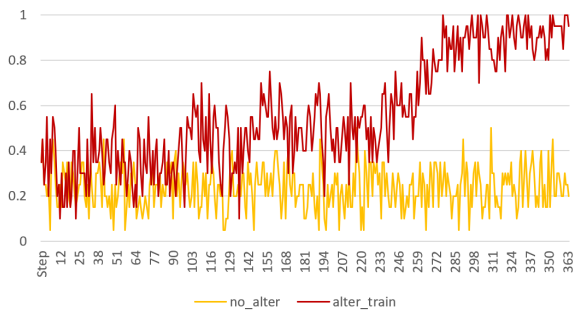


Fig. 12. accuracy improvement with or without alternatively training

VI. Discussions

This work examined the idea that active touching can indeed improve the classification prediction accuracy with only tactile sensors input. And there are some further directions that we could explore. Since the active touching policy did improve the quality of classification information, whether such a training can also improve the quality of gathering information for other usage, for instance, the information to generate better grasping policy or being used for 3D reconstruction.

VII. Acknowledgement

I would like to thank Dr. David Held for the guidance and support for this project. I'd like to thank Rachel Burcin and Dr. John Dolan for their effort in organizing this excellent program.

References

[1] Justin Lin, Roberto Calandra, and Sergey Levine. Learning to identify object instances by touch: Tactile recognition via multimodal matching. *CoRR*, abs/1903.03591, 2019.

[2] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada. Learning to grasp with primitive shaped object policies. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 468–473, Jan 2019.

[3] Sascha Fleer, Alexandra Moringen, Roberta L. Klatzky, and Helge J. Ritter. Learning efficient haptic shape exploration with a rigid tactile sensor array. *CoRR*, abs/1902.07501, 2019.

[4] Dinesh Jayaraman Justin Lin Wenzhen Yuan Jitendra Malik Edward H. Adelson Sergey Levine Roberto Calandra, Andrew Owens. More than a feeling: Learning to grasp and regrasp using vision and touch. *CoRR*, abs/1805.11085, 2018.

[5] Krishnan Srinivasan Parth Shah Silvio Savarese Li Fei-Fei Animesh Garg Jeannette Bohg Michelle A. Lee, Yuke Zhu. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. *CoRR*, abs/1810.10191, 2018.

[6] Mayur Mudigonda, Pulkit Agrawal, Michael Deweese, and Jitendra Malik. Investigating deep reinforcement learning for grasping objects with an anthropomorphic hand, 2018.

[7] Roberto Calandra, Andrew Owens, Manu Upadhyaya, Wenzhen Yuan, Justin Lin, Edward H. Adelson, and Sergey Levine. The feeling of success: Does touch sensing help predict grasp outcomes? *CoRR*, abs/1710.05512, 2017.

[8] Pietro Falco, Shuang Lu, Andrea Cirillo, Ciro Natale, Salvatore Pirozzi, and Dongheui Lee. Cross-modal visuo-tactile object recognition using robotic active exploration. 05 2017.

[9] Adithyavairavan Murali, Yin Li, Dhiraj Gandhi, and Abhinav Gupta. Learning to grasp without seeing. *arXiv preprint arXiv:1805.04201*, 2018.

[10] Ehud Rivlin and Héctor Rotstein. Control of a camera for active vision: Foveal vision, smooth tracking and saccade. *International Journal of Computer Vision*, 39(2):81–96, 2000.

[11] D. Jayaraman and K. Grauman. End-to-end policy learning for active visual categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1601–1614, July 2019.

[12] Ricson Cheng, Ziyan Wang, and Katerina Fragkiadaki. Geometry-aware recurrent neural networks for active visual recognition. In *Advances in Neural Information Processing Systems*, pages 5081–5091, 2018.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[14] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, Nov 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[16] Colin M. Light, Paul H. Chappell, and Peter J. Kyberd. Establishing a standardized clinical assessment tool of pathologic and prosthetic hand function: normative data, reliability, and validity. 83(6):776–783.

[17] Modular prosthetic limb shap test suites 1.31.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

RISS | ROBOTICS INSTITUTE SUMMER SCHOLARS