INTEGRATING PERCEPTION, PREDICTION AND CONTROL FOR ADAPTIVE MOBILE NAVIGATION

by

Kapil D. Katyal

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

March, 2021

Abstract

Mobile robots capable of navigating seamlessly and safely in pedestrian rich environments promise to bring robotic assistance closer to our daily lives. A key limitation of existing navigation policies is the difficulty to predict and reason about the environment including static obstacles and pedestrians. In this thesis, I explore three properties of navigation including prediction of occupied spaces, prediction of pedestrians and measurements of uncertainty to improve crowd-based navigation. The hypothesis is that improving prediction and uncertainty estimation will increase robot navigation performance resulting in fewer collisions, faster speeds and lead to more socially-compliant motion in crowds.

Specifically, this thesis focuses on techniques that allow mobile robots to predict occupied spaces that extend beyond the line of sight of the sensor. This is accomplished through the development of novel generative neural network architectures that enable map prediction that exceed the limitations of the sensor. Further, I extend the neural network architectures to predict multiple hypotheses and use the variance of the hypotheses as a measure of uncertainty to formulate an information-theoretic map exploration strategy. Finally, control algorithms that leverage the predicted occupancy map were developed to demonstrate more robust, high-speed navigation on a physical small form factor autonomous car.

I further extend the prediction and uncertainty approaches to include modeling pedestrian motion for dynamic crowd navigation. This includes developing novel techniques that model human intent to predict future motion of pedestrians. I show this approach improves state-of-the-art results in pedestrian prediction. I then show errors in prediction can be used as a measure of uncertainty to adapt the risk sensitivity of the robot controller in real time. Finally, I show that the crowd navigation algorithm extends to socially compliant behavior in groups of pedestrians.

This research demonstrates that combining obstacle and pedestrian prediction with uncertainty estimation achieves more robust navigation policies. This approach results in improved map exploration efficiency, faster robot motion, fewer number of collisions and more socially compliant robot motion within crowds.

Primary Reader and Advisor: Professor Gregory D. Hager

Secondary Readers: Professor Chien-Ming Huang, Dr. Philippe Burlina

Acknowledgments

There have been many individuals from both JHU and APL that have been instrumental in helping and guiding me along the PhD process. First and foremost, I would like to thank my advisor, Professor Greg Hager for his many insights along the way. In spite of his many duties, he's always able to quickly understand the nuances of any particular problem you are working and provide critical and insightful feedback and direction. Over the years, he's played a key role in helping me think about how to approach a problem and suggest the right set of experiments or ablation studies that will effectively communicate the impact of the results. I also want to thank Professor Chien-Ming Huang who has helped me guide the research towards dynamic obstacles and dealing with the complex challenges of navigating in the presence of pedestrians. Professor Austin Reiter was also a great resource for brainstorming ideas. He even joined some of the APL projects and was a lot fun to work with. Professor Marin Kobilarov has also been extremely helpful in thinking about how to operationalize the research into impactful demonstrations. I also want to thank additional members from LCSR who agreed to be on my GBO committee. These include Professor Mehran Armand and Professor Gregory Chirikjian for taking time out of their busy schedules to help serve on the committee.

In addition to the JHU faculty, I would like to thank many of the students that I have interacted with during my time. These students include Dr. Chris Paxton, Andrew Hundt, Yuxiang Gao, Dr. Colin Lea, Molly O'Brien, Jin Bai, Dr. Shahriar Sefati, and Dr. Subhransu Mishra. It was a lot fun getting to work with you and I look forward to seeing all of the great things you will do in your careers.

On the APL side, there are a number of people who have had a major contribution to my career development. First and foremost, I would like to thank Dr. I-Jeng Wang and Dr. Phil Burlina. Both have been great in terms of supporting my research, while providing guidance and mentoring along the way. In addition, I would like to thank the Research and Exploratory Development management for their unconditional support of pursuing a PhD while working full time by providing internal research and development funding and flexible work schedules. This includes Dr. Andrew Merkle, Dr. Jim Schatz, John Bigelow, Bart Paulhamus, David Patrone, Dr. Chris Ratto, Dr. Brock Wester, Ashley Llorens, Dr. Reed Young, Dr. Mike Wolmetz, Dr. Robert Osiander and Bob Bamberger.

The technical staff at APL has also been an amazing group of people to work with. They include Dr. Matt Johannes, Bobby Armiger, Dr. Mike Kutzer, Chris Brown, Matt Para, Aseem Raval, Andrew Harris, Dr. Joe Moore, Dr. Kevin Wolfe, Dr. Ryan Murphy, Dr. Katie Popek, Ted Staley, Dr. Corban Rivera, Dr. Galen Mullins, Matt Rich, Adam Polevoy, Craig Knuth, Dr. Jared Markowitz, Dr. Seth Billings, Dr. Aurora Schmidt, Dr. Matt Fifer, Dr. Francesco Tenore, Dr. Grace Hwang, and many others.

There have also been many official and unofficial mentors at APL that have been instrumental in guiding my career. These include Mike McLoughlin for providing so many opportunities throughout the years, Dr. James Beaty for his ability to lead any team around the globe, Dr. Eddie Tunstel who embodies technical competence and humility and Michelle Chen for trusting her instincts which gave me an opportunity to start down this path. Thank you for your guidance and wisdom over the years.

And of course, none of this would be possible without my family. Thanks to my sister and her husband for always being there with support. My sisters, mother, and father-in-law have also been incredibly encouraging throughout the years. Thank you to my late father and my mother who sacrificed leaving their family behind in search of a better future for their kids.

Finally, thank you to my wife, Nidhi and my kids, Emily, Devan and Dylan for their unconditional support and patience over the years.

Dedication

This thesis is dedicated to my wife who has been a constant source of encouragement, support, wisdom and patience. During times of stress, you would always find a way to put things in perspective and find the silver lining. Thanks for all that you have done for me and the kids throughout the years. We are truly lucky to have you in our lives.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The expectation is that robots and humans will coexist with one another in a seamless manner to improve the quality of life and bring robotic assistance to the mainstream. A critical component to achieve this goal is the ability for mobile robotic systems to navigate safely and effectively in new, unstructured environments consisting of crowds of pedestrians. This requires several capabilities including the ability to map the environment, generate collision free paths to the desired goal, and navigating to the goal while avoiding static and dynamic obstacles.

There has been substantial work in adaptive robot navigation that will be further elaborated in Chapter 2. In spite of past research, many limitations still exist that prevent mobile robots from navigating efficiently and at high speeds with human-like ability. Hard coded navigation policies work well in the intended domain but are difficult to generalize to new environments and pedestrian behavior patterns. Robots typically operate based on the direct line-of-sight and field-of-view of the sensors on board the robot. This fundamentally limits the ability to plan at high speeds without a predictive model. Finally, robot motion in crowded environments tends to be risk averse and lacks social norms that humans have acquired over years of navigating around other humans.

As part of this thesis, I explore new capabilities that improve adaptive mobile robot navigation. Specifically, the objective is to integrate (1) perception including reasoning about the environment and pedestrians, (2) prediction, including prediction of static and dynamic obstacles and (3) control algorithms that leverage perception, prediction and uncertainty to improve navigation performance.

1.1 Motivation

In this section, I describe a few motivating examples that provide context for the research presented in this thesis.

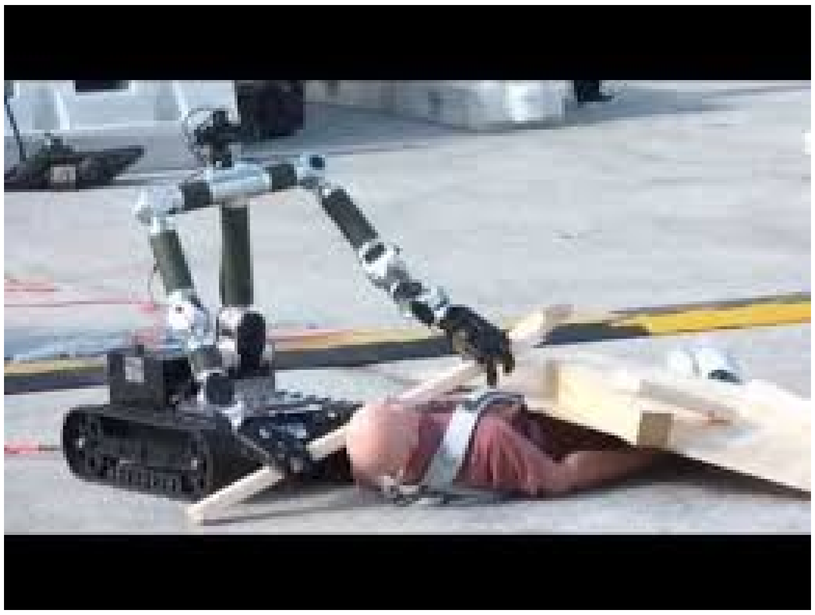


Figure 1.1: An example of a mobile manipulator system for disaster response.

1.1.1 Search and Rescue Robots

A significant opportunity for mobile robots to make a critical contribution lies in search and rescue robotics. This is an area that has been studied at great depths in the past [10] however we have yet to see robots play a key role in aiding humans in the disaster response. A recent example includes the 2011 Fukushima Daiichi nuclear disaster caused by an earthquake and subsequent tsunami resulting in over 500 deaths. Scenarios such as this necessitate leveraging robotic systems to perform search and rescue tasks in order to find potential survivors quickly and efficiently. The few robots that were deployed include the iRobot Warrior and Packbot. While compact and agile, these robots are traditionally teleoperated, requiring significant human intervention

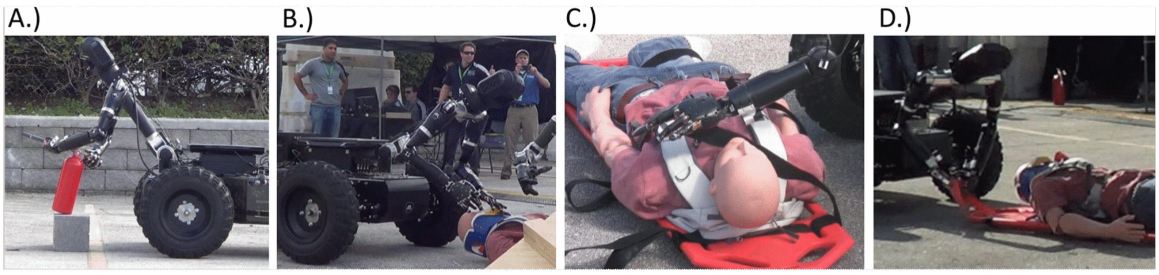


Figure 1.2: Mobile manipulation for disaster response A.) Manipulate a fire extinguisher B.) Find and navigate to a spinal cord injury victim C.) Place the victim securely on a spineboard and D.) Quickly relocate the victim a safe distance away.

to perform search and rescue tasks. In Figures 1.1 and 1.2, I show examples of more advanced search and rescue robots developed by JHU/APL in the past to aid in the response [11, 12]. While these mobile manipulator systems have demonstrated capabilities to improve disaster response, an opportunity exists to develop better autonomy that allows robots to navigate faster, explore spaces more efficiently and navigate through pedestrians with ease. The culmination of this research has the potential to significantly improve search and rescue capabilities that exist today.

1.1.2 Crowd Navigation

The second motivating scenario deals with crowd navigation. This is an important capability for many service-oriented robotic systems including delivery robots, warehouse robot systems, and robotic tour guides. One of the main challenges is dealing with high density pedestrian traffic as in Fig. 1.3. This scenario presents a challenging navigation task as the robot must move through the crowd while avoiding collision with other pedestrians and obstacles. Many of the existing works use explicit, hand-coded policies that are error prone. In addition, while navigation may be functionally correct, it remains difficult to capture socially normal behaviors that enable acceptance of robotic systems into society. Some navigation methods focus on predicting future pedestrian motion to improve navigation. However, these approaches typically predict motion based on past trajectories as opposed to inferring internal states such as intent. Finally, a critical aspect that has been overlooked in the literature is the consideration of dynamic social groups (Fig. 1.4). The ability to model and detect dynamic social groups to enable socially appropriate robot navigation has the potential to influence people's perceptions of trust and acceptance of mobile robots embedded in human environments. In conclusion, there remains a significant opportunity to further improve predictive capabilities, develop risk-sensitive control algorithms as well as learning based policies that simultaneously improve navigation performance while also capturing socially normal behaviors.

Figure 1.3: Example of a crowded intersection representing a challenging navigation task.

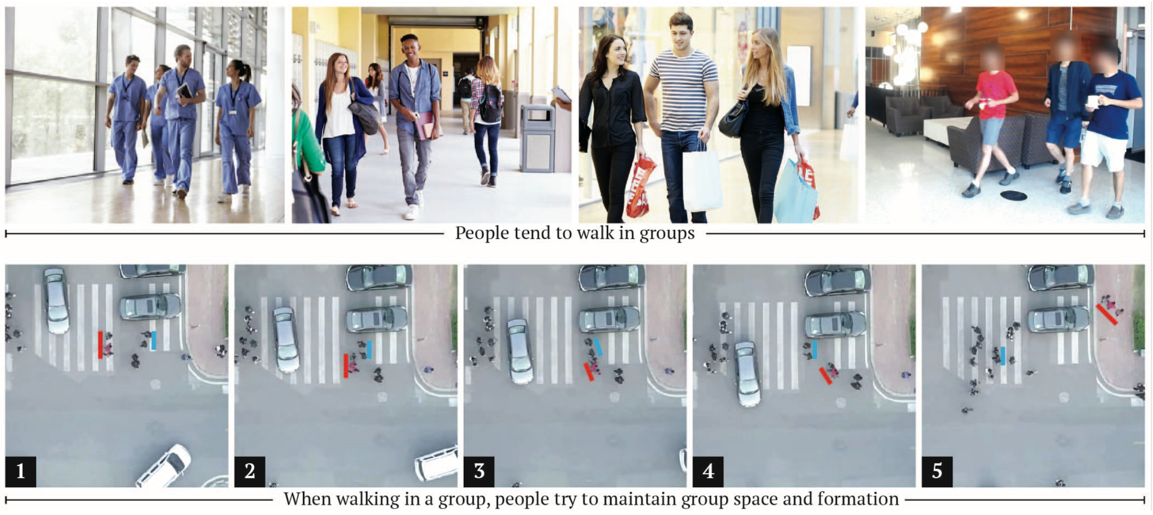

People tend to walk in groups
1
2
3
4
5
When walking in a group, people try to maintain group space and formation

Figure 1.4: Dynamic social groups in naturalistic settings. People tend to walk in groups (Top) and try to maintain group space and their formation (blue and red lines) during walking (Bottom).

1.2 Outline

The major goal of this work is to develop novel techniques that advance predictive capabilities with uncertainty to improve robot navigation performance in the presence of crowds of pedestrians. Traditional methods of prediction primarily operate on low dimensional state vectors. The approach focuses on developing techniques that can operate on and generate high-dimensional representations such as raw camera observations and occupancy maps. In Chapter 2, I provide background information, describe existing approaches and discuss the main limitations.

In Chapter 3, I focus on developing techniques that can make predictions of future occupied spaces that are generated from existing mapping techniques. Here, I assume the ability to generate a map that represents occupied, unoccupied and free space in the environment based on the current sensor readings on the robot. I first investigate various neural network architectures and loss functions to determine the types of generative networks and optimization techniques that produce the most accurate predicted occupancy maps. I then extend the architectures to produce an uncertainty metric. By leveraging prediction with uncertainty, I demonstrate the ability to improve the efficiency of exploring new environments compared to several baseline exploration approaches. Second, a controller that leverages the predicted maps was developed which demonstrates the ability to travel at higher speeds with fewer collisions using a physical small platform race car.

In Chapter 4, I shift the focus to dynamic obstacles, in particular pedestrians. This is accomplished through techniques that perform state estimation and uncertainty of pedestrians using high-dimensional observations as the input. By applying traditional Kalman Filter techniques on latent embeddings representing the pedestrian, the state and uncertainty estimation is improved against state-of-the-art baselines while also reducing the computational requirements. I then focus on predicting future motion of pedestrians. I show that prediction accuracy can be improved by reasoning about the pedestrian's intent compared to state-of-the-art baselines using traditional pedestrian datasets. Finally, an adaptive reinforcement learning (RL) policy was developed that leverages the predicted pedestrian motion to improve navigation performance while reducing the number of collisions compared to baseline policies.

In Chapter 5, I extend the RL approaches developed in Chapter 4 to improve navigation performance by incorporating group information as part of the network policy. By taking into consideration the group dynamics, significant improvements in navigation performance and socially compliant behavior are demonstrated within crowds of pedestrians.

Finally, Chapter 6 summarizes the main contributions, discusses the limitations and provides an overview of future work that extend the main ideas described in this thesis.

1.3 Thesis Statement

I hypothesize that high-dimensional generative neural networks can learn to predict with uncertainty future static and dynamic obstacles in the environment. Further, I posit that high-dimensional prediction with uncertainty and modeling pedestrian properties such as intent and group membership will improve adaptive robot navigation policies resulting in more efficient exploration of new environments, navigation at faster speeds, reduced number of collisions and improved social normal behaviors in crowds of pedestrians.

1.4 Contributions

The research ideas presented in this thesis have been decomposed into occupancy map prediction, pedestrian prediction and group aware navigation. This work has been published in a series of conference papers including:

1. Chris Paxton, Yotam Barnoy, **Kapil D. Katyal**, Raman Arora, Gregory D. Hager. Visual Robot Task Planning. ICRA 2019: 8832-8838.

2. **Kapil D. Katyal**, Katie M. Popek, Chris Paxton, Philippe Burlina, Gre-

gory D. Hager. Uncertainty-Aware Occupancy Map Prediction Using Generative Networks for Robot Navigation. ICRA 2019: 5453-5459.

3. **Kapil Katyal**, Gregory Hager and Chien-Ming Huang. Intent-Aware Pedestrian Prediction for Adaptive Crowd Navigation. ICRA 2020.

4. **Kapil Katyal**, I-Jeng Wang, and Gregory D. Hager. Out-of-Distribution Robustness with Deep Recursive Filters. ICRA 2021 (Accepted).

5. **Kapil Katyal**, Adam Polevoy, Joseph Moore, Craig Knuth, Katie M. Popek. High-Speed Robot Navigation using Predicted Occupancy Maps. ICRA 2021 (Accepted).

6. **Kapil Katyal**, Yuxiang Gao, Jared Markowitz, I-Jeng Wang, and Chien-Ming Huang. Group-Aware Robot Navigation in Crowded Environments. RA-L (Submitted for review on 12/21/2020).

7. **Kapil Katyal**, Katie Popek, Chris Paxton, Joseph Moore, Kevin Wolfe, Philippe Burlina, Gregory D. Hager. Occupancy Map Prediction Using Generative and Fully Convolutional Networks for Vehicle Navigation. arXiv:1803.02007.

8. **Kapil Katyal**, Katie Popek, Gregory D. Hager, I-Jeng Wang, Chien-Ming Huang. Prediction-Based Uncertainty Estimation for Adaptive Crowd Navigation. Human Computer Interaction-I 2020.

9. **Kapil Katyal**, I-Jeng Wang, Gregory Hager and Chien-Ming Huang. Intent-Aware Human Motion Prediction using Deep Generative Neural Networks. UMD Do Good Robotics Symposium 2019.

Other publications that inspired the research presented in this thesis include the following:

1. **Kapil D Katyal**, Christopher Y Brown, Steven A Hechtman, Matthew P Para, Timothy G McGee, Kevin C Wolfe, Ryan J Murphy, Michael DM Kutzer, Edward W Tunstel, Michael P McLoughlin, Matthew S Johannes. Approaches to robotic teleoperation in a disaster scenario: From supervised autonomy to direct control. IROS 2014: 1874-1881.

2. Joseph Moore, Kevin C Wolfe, Matthew S Johannes, **Kapil D Katyal**, Matthew P Para, Ryan J Murphy, Jessica Hatch, Colin J Taylor, Robert J Bamberger, Edward Tunstel. Nested marsupial robotic system for search and sampling in increasingly constrained environments. 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC): 002279-002286.

3. Edward W. Staley, **Kapil D. Katyal**, Philippe Burlina. DRL Based Intelligent Joint Manipulator and Viewing Camera Control for Reaching Tasks and Environments with Obstacles and Occluders. IJCNN 2018: 1-7.

4. **Kapil D. Katyal**, Edward W. Staley, Matthew S. Johannes, I-Jeng Wang., Austin Reiter, and Phil Burlina. In-hand robotic manipulation via deep reinforcement learning. 2016 NeurIPS Workshop on Deep Learning for Action and Interaction. December 2016.

5. **Kapil D. Katyal**, I-Jeng Wang, Philippe Burlina. Leveraging Deep Reinforcement Learning for Reaching Robotic Tasks. CVPR Workshops 2017: 490-491.

Chapter 2

Background

The contents of this thesis span several topics in the robotics and machine learning literature. This chapter provides a brief introduction to these topics as well several references for further background information. Specifically related to robot navigation, this chapter covers introductory material including robot localization, mapping, state estimation, and pedestrian prediction techniques. Further, I provide a background on the machine learning techniques used in later chapters of the thesis including deep learning, generative neural networks, and reinforcement learning. Finally, I describe several existing datasets used in subsequent chapters as training data and evaluation against existing approaches.

2.1 Robot Localization and Mapping

Simultaneous localization and mapping (SLAM) algorithms are critical components to robot navigation that allow the robot to simultaneously build a map of the environment and localize itself within the map. A fundamental source for theory and algorithms related to SLAM algorithms can be found in Principles of Robot Motion by Choset et al. [13]. More formally defined, the objective of the SLAM algorithm is to estimate the robot state, x_t and map of the environment, m_t given sensor observations, o_t and control signals u_t over a discrete time horizon, t. The objective of the SLAM algorithm is to compute:

$$P(m_{t+1}, x_{t+1} | o_{1:t+1}, u_{1:t})$$

. using the following update equations:

$$P(x_t|o_{1:t}, u_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t|x_t, m_t, u_{1:t}) \sum_{x_{t-1}} P(x_t|x_{t-1} P(x_{t-1}|m_t, o_{1:t-1}, u_{1:t})/Z$$

$$P(m_t|x_t, o_{1:t}, u_{1:t}) = \sum_{x_t} \sum_{m_t} P(m_t|x_t, m_{t-1}, o_t, u_{1:t}) P(m_{t-1}, x_t|o_{1:t-1}, m_{t-1}, u_{1:t})$$

In this thesis, I focus primarily on vision based navigation that operates on multimodal sensor data including RGB cameras, LIDAR and Inertial Measurement Units (IMU) to generate an occupancy map of the environment. An occupancy map is a grid-like metric structure where each cell of the grid represents the posterior probability of that space being occupied by an obstacle. In addition to mapping the environment, the robot must localize itself within the generated map. This problem is referred to as the SLAM problem. There are many SLAM algorithms that exist and are summarized by several survey papers [14, 15]. In this thesis, I primarily use two SLAM algorithms, Google Cartographer [16] and Real-Time Appearance-Based Mapping (RTAB-Map) [17]. Cartographer works by combining local and global approaches to develop the SLAM algorithm. The local approach consists of consecutive scans of the environment to generate submaps consisting of smaller segments of the global map. Global maps are then combined using loop closure detection and graph optimization techniques to produce high resolution maps of the environment. RTAB-Map uses a different approach to mapping and relies heavily on matching appearance based features across frames. Similar to Cartographer, RTAB-Map also uses loop closure and graph optimization, however uses a bag-of-words approach to perform the loop closure step. In this method, RTAB-Map extracts visual features from the environment and maps them to a quantized visual word vocabulary. These visual words are then used to perform the feature to feature matching across frames of images to construct the map.

In this thesis, I primarily use Google Cartographer to perform mapping when using a LIDAR sensor as the input and the RTAB-Map algorithm when using stereo cameras such as the Intel® Realsense camera. These frameworks

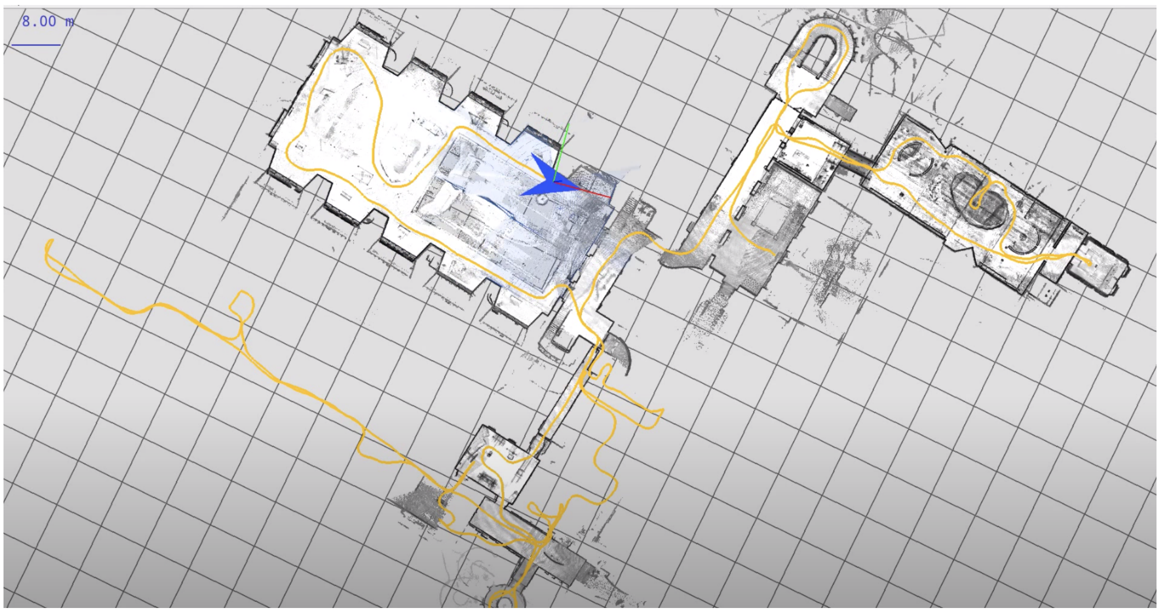


Figure 2.1: Sample occupancy map generated from Google Cartographer SLAM package

provide the occupancy map data structure that is used extensively in Chapter 3 to provide the input and output occupancy maps for predicting occupied regions beyond the sensor's line of sight.

2.2 State and Uncertainty Estimation

State and uncertainty estimation are also critical components related to navigation. To avoid static and dynamic obstacles in the environment, the robot must estimate the state of obstacles in the scene to develop collision free paths towards the goal. In this thesis, I primarily rely on a family of recursive filters for state estimation, namely the Kalman filter [18], also known as the Kalman-Bucy filter. The Kalman filter is a state estimation algorithm used in many application including guidance, navigation and control (GNC), time series-based signal processing, and robotic motion planning and controls. It operates by receiving a series of measurements containing statistical noise over periods of time and produces estimates of unknown variables that account for the errors in addition to an estimate of uncertainty. While this work focuses primarily on the Kalman filter, this method represents a family of algorithms called recursive Bayesian filtering that assumes Markov state dynamics and observation models. The general form of the Kalman filter is a recursive solution to linear filtering and use the following two equations for the state dynamics and observation models:

$$\text{State Dynamics Model: } \hat{x}_{t+1} = f(x_t, u_t, w_t)$$

$$\text{Observation Model: } \hat{o}_t = h(x_t, v_t)$$

In the classic sense of the Kalman filter, $\hat{x}_{t+1}$ and $\hat{o}_t$ are governed by linear models however extensions of the Kalman filter include the Extended Kalman filter which allows for non-linear state dynamics and measurement models by linearizing the two equations.

There are two main steps related to the Kalman filter algorithm, the state prediction step followed by the state update step. During the prediction step, the algorithm produces a predicted, a priori estimate of the state and covariance matrices as propagated by the provided dynamics model. The update step uses the observation along with the a priori predicted state to produce an updated a posterior state estimation and covariance. More formally, the state prediction steps are governed by the following equations.

$$\hat{\mathbf{x}}_t^{'} = \hat{\mathbf{F}}_t \hat{\mathbf{x}}_{t-1} + \mathbf{B}_t \mathbf{u}_t$$

$$\hat{\mathbf{P}}_t^{'} = \hat{\mathbf{F}}_t \mathbf{P}_{t-1} \hat{\mathbf{F}}_t^T + \mathbf{Q}_t$$

The Kalman filter update step receives the a priori estimate along with a new observation to produce an a posteriori estimate according to the following equations.

$$\tilde{\mathbf{s}}_t = \mathbf{w}_t - \mathbf{H}_{\mathbf{t}} \hat{\mathbf{x}}_t^{'}$$

$$\mathbf{S}_t = \mathbf{H}_t \hat{\mathbf{P}}_t^{'} \mathbf{H}_t^T + \mathbf{R}_t$$

$$\mathbf{K}_t = \hat{\mathbf{P}}_t^{'} \mathbf{H}_t^T \mathbf{S}_t^{-1}$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^{'} + \mathbf{K}_t \tilde{\mathbf{s}}_t$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \hat{\mathbf{P}}_t^{'}$$

In these equations, $\hat{\mathrm{x}}_t^{'}$ and $\hat{\mathbf{P}}_t^{'}$ represent the a priori, predicted state and covariance estimation, F represents the dynamics model, H represents the observation model, Q is the process noise covariance matrix, R is the observation noise covariance matrix, and B is the optional control input model. Further, $\tilde{s}_t$ is the innovation, S_t is the covariance of the innovation and K is the optimal Kalman gain that balances the contributions of the predicted state and the innovation covariance. Finally, $\hat{x}_t$ and P_t represent the updated a posterior state and covariance estimate.

In this thesis, I focus on leveraging state estimation techniques for detecting and predicting pedestrian motion. Specifically, in Chapter 4, I extend traditional Kalman filter techniques to produce a posterior state and uncertainty estimation of a latent feature embedding representing detected pedestrians in the scene. I demonstrate this approach leads to more robust state and uncertainty estimation under noisy observations.

2.3 Pedestrian Prediction Techniques

One of the major challenges in robot navigation is to develop trajectories that safely avoid pedestrians as well as minimizes potential discomfort assessed by pedestrians. Many existing works treat pedestrians as a moving obstacle using simple linear dynamics to model and predict future motion based on past observations. Formally, this problem can be stated as follows: at given

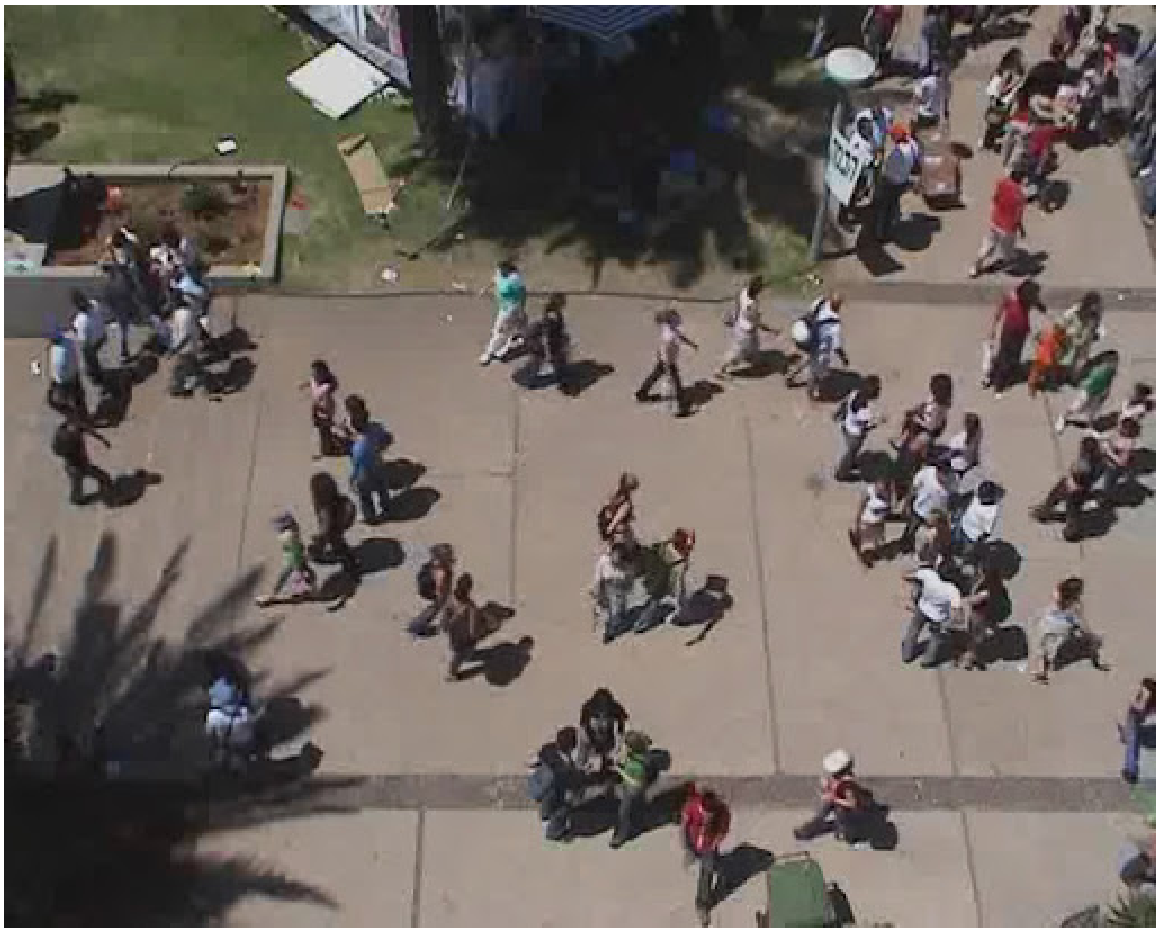


Figure 2.2: Sample image from UCY pedestrian dataset [1]

time t, the state of the pedestrian i can be defined as $X_i^t = (x_i^t, y_i^t)$. The pedestrian state is observed during a time window $t = 1$ to $t = T_{obs}$ represented as $X_i^{obs} = [(x_i^1, y_i^1), ..., (x_i^{T_{obs}}, y_i^{T_{obs}})]$. The objective is to predict the pedestrian state from time $t = T_{obs+1}$ to $t = T_{pred}$ represented as:

$$Y_i^{pred} = [(x_i^{T_{obs+1}}, y_i^{T_{obs+1}}), ..., (x_i^{T_{pred}}, y_i^{T_{pred}})]$$

. One of the major challenges of using a simple linear model to predict future motion is that pedestrian motion is often nonlinear and is typically motivated by an underlying goal or intent that is frequently unobservable.

Many works have begun using deep neural networks to model the nonlinear pedestrian motion with great success including [19–22]. These approaches typically learn a latent representation describing a spatial and temporal feature vector that is subsequently used to generate future predictions. These approaches are typically evaluated against two widely used, publicly available datasets—ETH [23] and UCY [1]—consisting of 5 unique datasets (ETH, Hotel, Univ, Zara1, and Zara2) with 4 scenes. The datasets include pedestrian motion with a top down view and annotated pedestrian positions with respect to the world frame. The metrics used for evaluation include the average displacement error (ADE) and final displacement error (FDE). The ADE (in meters) is the L2 distance between the ground truth and predicted pedestrian trajectories for each trajectory point. The FDE (in meters) is the final displacement distance between the last point in the predicted trajectory and the ground truth.

In Chapter 4, I describe existing approaches to pedestrian prediction by estimating intent. Further, I augment existing latent representations with the intent estimation to improve ADE and FDE using the datasets described above.

2.4 Deep Neural Networks

Deep learning is a class of machine learning techniques that leverage artificial neural networks (ANNs) to learn feature representations used for classification, regression or other machine learning tasks. ANNs have been loosely derived from biological networks where each neuron can transmit information to other neurons. These biological neurons are arranged in layers where different layers perform different biological tasks. ANNs model biological neural networks with a composition of artificial neurons that consist of an input and output signal to other neurons. The output of a neuron consists of weighting the input signal and adding a bias term followed by an activation function that allows non-linearity between layers. With deep learning, this concept is extended to multiple deep layers of ANNs, where each layer represents an extracted feature embedding passed into subsequent layers. The final layer is then used to perform the machine learning task.

There are a variety of layers that can be included that process the input signal in different ways. In this work, I primarily focus on three types of layers, fully connected, convolutional and recurrent layers. A fully connected layer is one where each neuron is connected to all neurons on the subsequent layer. This creates a flattening effect that maps the input dimension to the desired output dimension of the layer. A convolutional layer is typically used in inputs that contain spatial similarity such as images. Convolutional layers are specified by a kernel size (height x width) and number of input and output channels. The convolutional layer convolves the input values using the specified kernel by computing the dot product between the input window and filter entries. The main advantage of using a convolutional layer is reduced number of free parameters by assuming spatial similarity and structure across the kernel size. Finally, recurrent layers, such as long, short term memory are typically used to capture temporal sequences found in time series data. A recurrent network differs from a traditional feedforward network by not only considering the current input but also information received from previous time steps via internal memory cells. Recurrent neural networks have significantly improved many machine learning tasks with temporal structure such as natural language processing, translation and video activity recognition. In this work, I rely heavily on these three forms of network layers to process input images, change the dimensionality of the feature embedding and capture both spatial temporal

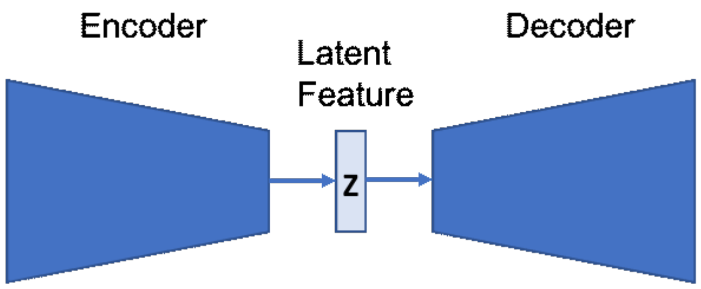


Figure 2.3: Autoencoder Architecture

information when learning features.

2.5 Deep Generative Neural Networks

Deep generative neural networks are a specific class of deep learning that is a combination of generative models and deep neural networks. The main goal of deep generative models is to learn the true underlying distribution of the data so that new data points can be generated. Deep generative models can be thought of as an extension of classic unsupervised learning techniques including principal component analysis (PCA), clustering and mixture model techniques.

A classical deep generative model is the auto-encoder [24] which consists of encoder and decoder networks and a compressed, bottleneck later in between as depicted in Fig. 2.3. Typically, this style of a neural network attempts to reconstruct the input during the optimization by minimizing the mean squared error (MSE) reconstruction loss between the output of the neural network and

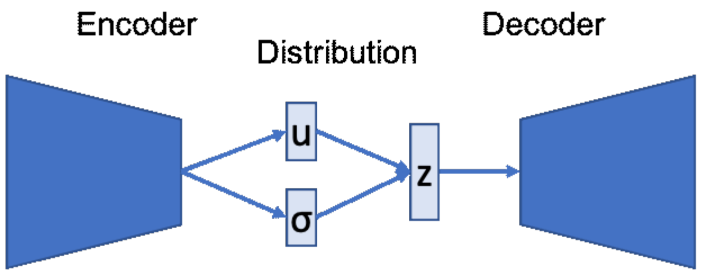


Figure 2.4: Variational Autoencoder Architecture

the input image. Using this approach, the learned latent feature found in the bottleneck layer is used as a lower dimensional feature vector that represents the high dimensional input.

A second class of generative networks are known as variational autoencoders (VAEs) [25]. In VAEs, the main objective is to learn a distribution representing the high dimensional input instead of a latent feature vector directly as shown in Fig. 2.4. By learning parameters of a distribution, the distribution can be sampled many times to generate diverse samples representing the uncertainty of the generation process.

In this thesis, I primarily use these variants of generative neural networks for produce predicted high dimensional occupancy maps based on the limited FOV occupancy grid. I leverage VAE techniques to capture a distribution of the prediction that can be sampled to generate a population of hypothetical predictions. In Chapter 3, I show these networks are capable of producing highly accurate predicted occupancy maps using customized architectures and loss functions specific to the robotic navigation domain.

2.6 Reinforcement Learning

Reinforcement learning (RL) has been widely used to allow agents (i.e. robots) to learn how to interact with the environment and is covered in great depth by Sutton and Barto [2]. Traditionally, RL can be formalized as a Markov Decision Process (MDP), as summarized in Fig. 2.5, which consists of an *agent* interacting with the *environment*. The agent selects *actions* that result in a *reward* causing a change to the *environment* as observed as the next *state*. Q-Learning is a form of RL where the primary objective is to develop a policy that allows the *agent* to select an *action* given the current *state* in order to maximize the expected *reward*. The Q-value correlates to the quality of choosing the action given the state and is iteratively updated by Eq. 2.1.

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)] \quad \textbf{(2.1)}$$

where $Q_{t+1}(s_t, a_t)$ represents the updated Q-value, $Q_t(s_t, a_t)$ is the previous Q-value, $\alpha_t(s_t, a_t)$ represents the learning rate, R_{t+1} is the immediate reward, γ represents the discount factor, and $\max_a Q_t(s_{t+1}, a)$ represents the estimate of the optimal future value. In the simplest case, the Q-value is stored in a table with the rows and columns corresponding to the states and actions respectively.

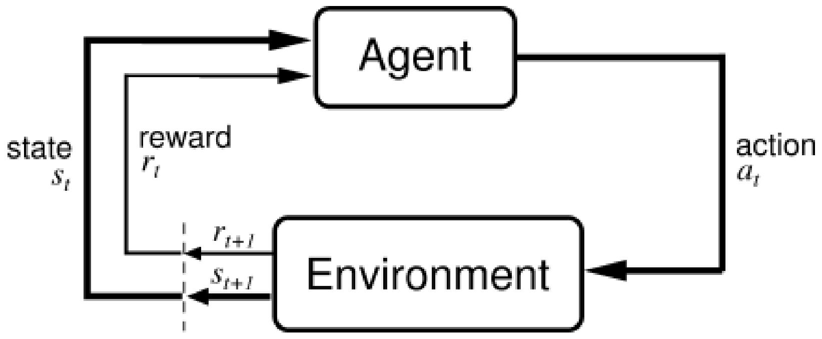


Figure 2.5: Reinforcement Learning based on MDP [2]

Using this definition of the Q-value, the selected action is defined by Eq. 2.2

$$\pi(s) = \arg\max_a(Q(s,a)) \tag{2.2}$$

With Q-Learning, typically assumptions of the system state are made for computational efficiency. In the case of the Atari 2600 game Breakout, the paddle position, ball position, and block locations could all be used to represent the state. For manipulation, the state of the system could be modeled as the Denavit-Hartenberg (DH) parameters defining the robot kinematics, the current joint angles and the location of the target object. The issue with this approach is that the state is specific to the application and would not apply to other games or manipulators. The preferred approach would be to model the state using raw image pixels of the scene. This would allow maximum generalization however, the challenge is the curse of dimensionality. Assuming the state is represented as 4 frames of an 84×84 pixel grayscale image, the state space is equal to $256^{4\times84\times84} \approx 10^{67970}$ dimensions. The magnitude of the state space would make convergence infeasible using modern computing platforms. Instead of using a table to represent the Q-values, a deep neural network (DNN) is used as a function approximator that maps raw image pixels to Q-values corresponding to potential actions, known as deep reinforcement learning (DRL) [26]. Using this approach, deep Q-learning has been able to provide general solutions to a wide variety of challenging problems using high dimensional inputs as the observation.

While deep Q-learning is able to approximate the Q function using a deep neural network, an alternative approach is to optimize the policy directly using policy gradients. In practice, policy gradients have demonstrated greater success on a wider variety of problems.

Another class of RL algorithms explicitly optimize the policy directly instead of estimating the action value function as in the case of the DQN architecture. Examples of this approach include Trusted Region Policy Optimization (TRPO) [27] and Proximal Policy Optimization (PPO) [28]. The motivation behind TRPO and PPO is to bound the improvement step during training to prevent learning in unstable regions. TRPO accomplishes this by bounding the KL-divergence between two distributions where as PPO uses computationally simpler first order methods as an approximation the bounded operation. In practice, PPO is simpler to implement and empirically performs as well or exceeds TRPO. In Chapters 4 and 5, I leverage the PPO algorithm as the fundamental RL algorithm to demonstrate crowd and group based navigation.

To summarize, in this chapter, I provide a brief introduction to important background topics related to robot navigation, state estimation, pedestrian prediction, deep neural networks and various reinforcement learning policies.

Chapter 3

Occupancy Map Prediction

3.1 Introduction

“Prediction is very difficult, especially if it’s about the future” – Niels Bohr. This is particularly true for robot navigation where robots are expected to travel with ease while encountering a wide variety of obstacles of varying shapes and sizes. In this chapter, I focus on developing high-dimensional occupancy map prediction techniques for static obstacles that allow a mobile robot to efficiently explore new environments and travel at high speeds. Traditional exploration strategies typically operate on the most recent sensor reading (e.g., lidar) to update an occupancy map corresponding to an internal representation of where obstacles exist in the environment, which are subsequently used by planning algorithms to generate a collision-free path to a target goal. A key limitations of existing approaches is that the planning horizon is limited to the field of view (FOV) of the sensor and do not take into account predictions that extend beyond the FOV. The underlying assumption is that informed exploration of new environments can be improved by reasoning about future occupied and unoccupied spaces.

This is evidenced by recent studies of human patterns of environment exploration that have shown that (i) labeling occupied spaces based on observations, (ii) making predictions of occupied spaces beyond line of sight and (iii) estimating the uncertainty in these predictions all contribute to efficient exploration of a new environment [29–31].

In this chapter, I lay a foundation that models human behavior with respect to informed exploration of new environments and high speed navigation. As a motivating example, if asked to explore a new environment, one might glance around the space, make predictions of occupied regions behind occluding obstacles and perhaps select a hallway to explore given that this space has the highest uncertainty. Following this intuition, I believe future predictions of occupancy maps can enable more efficient exploration strategies for mobile and aerial robotic systems. The approach to occupancy map prediction is similar to image completion [32, 33]. Similarly, by making future predictions of the environment, we should expect a more robust navigation policy allowing the robot to travel at faster speeds. To accomplish this, I leverage the fact that struc-

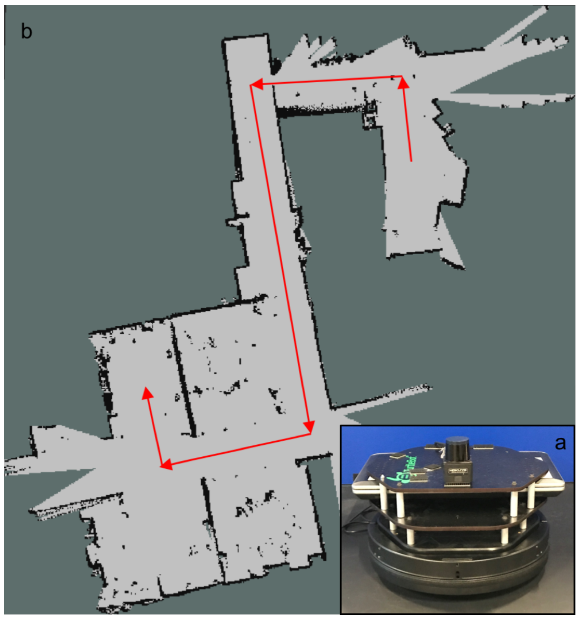


Figure 3.1: (a) Turtlebot robot used as platform for navigation, (b) Information-theoretic planning using uncertainty estimation as heuristic for exploration.

tural information from the observed geometry of the world can help us make useful predictions about the environment. These predictions are made by using variations of autoencoder networks which are capable of encoding a latent representation of images; these latent representations can be used to generate new examples of predicted spaces [34].

This chapter focuses on three main contributions. I explore neural network architectures that demonstrate the ability to generate future predictions of occupancy maps. I further show how the neural network architectures can be modified to make multiple hypotheses to provide a heuristic for map exploration. I then show how to leverage the predicted regions to improve the robustness of high speed robot navigation.

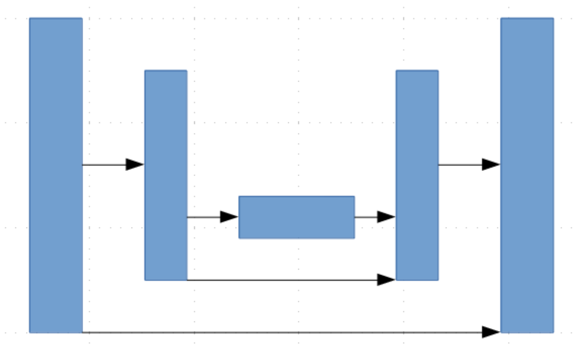


Figure 3.2: U-Net Neural Network Architecture with skip connections to bypass bottleneck layer

The organization and specific contributions of this chapter are the following:

- In Sec. 3.2, I briefly summarize prior work related to map prediction.
- In Sec. 3.3, I provide qualitative and quantitative comparisons of the prediction capabilities, performance and accuracy of various network architectures and loss functions.
- In Sec. 3.4, I extend the neural network architecture to capture the inherent uncertainty in robot navigation and use the uncertainty metric as a heuristic for an information-theoretic exploration strategy.
- Finally, in Sec. 3.5, a robot controller was developed that leverages the predicted map regions to achieve high-speed navigation using a physical RC car.

3.2 Prior Work

Deep Learning for Generative Models

Deep neural networks have been used in a number of promising ways to achieve high performance in domains such as vision, speech and more recently in robotics manipulation [35, 36]. Oh et al. used feedforward and recurrent neural networks to perform action-conditional video prediction using Atari games with promising results [37]. These have also been used in image completion, e.g., by Ulyanov et al. [38]. In addition, generative adversarial networks (GANs) have demonstrated a promising method for image generation [39]. Isola et al. proposed an approach for training conditional GANs, which create one image from another image [40]. While inpainting and image completion have made significant progress in recent years, they have yet to make a significant impact in the robotic navigation domain.

Map Exploration

Robotic navigation and specifically map exploration strategies have been studied extensively in the robotics literature. One of the seminal papers is frontier exploration by Yamauchi [3]. The key idea in this approach is to detect frontiers defined as borders between unexplored and open spaces of the environment. This work is further extended by Wirth et al. [6] who take into consideration the distance to the next frontier in the search strategy. Bai et al. propose an information theoretic approach to exploration [4, 5]. In this work, the authors use a combination of Bayesian optimization techniques and a Gaussian process to estimate mutual information throughout the robot's action space and select the action that minimizes the entropy of the map. Finally, O'Callaghan et al. [41] demonstrate using Gaussian processes as a statistical modelling technique for building occupancy maps by exploiting inherent structure of real-world environments.

Deep Learning for Navigation

More recently, several papers have described approaches to combine elements of deep neural networks with autonomous navigation. These include using deep neural networks for model predictive control [35]. Tamar et al. proposed Value Iteration Networks, which embed a planner inside a deep neural net architecture [42]. Several papers investigate the use of deep reinforcement learning to develop collision-free planning without the need of an internal map, however, these approaches are still restricted by the sensor's FOV without using a predictive model [43, 44].

High Speed Navigation

Adjacent to map prediction is the problem of planning the shortest path to a goal in an unknown environment. These methods perform some level of inference over the environment to inform motion planning. For example, in [45], they learn how to plan waypoints to a goal using a partial map of the environment. Both [46] and [47] use prior experience to reduce the size of viable environment hypotheses. Specifically, [46] learns the probability of collision of motion primitives whereas [47] utilizes experience-based map predictions in a belief space planner.

Elfhafsi *et al.* [48] incorporate map prediction with global path planning that respects the system dynamics, but only experimented in simulation.

Existing work has also considered applying reinforcement learning (RL) in tandem with occupancy maps or depth information to navigate an unknown environment. In [49], the method uses RGB images and predicts the probability and variance of collision while navigating towards a goal, but is incapable of global planning. In [50], the approach models the world as a POMDP and take an end-to-end approach to navigate to a goal. Additionally the method presented in [51] uses a partial map as inputs to a deep RL policy to navigate in unknown environments.

3.3 Neural Network Architecture Design

In this section, I formally define the problem of occupancy map prediction, describe the neural network architecture and loss functions used during training and introduce several baseline implementations used to compare relative performance to existing inpainting approaches. The main objective of this section is to understand which neural network architectures and loss functions provide the most accurate predictions of occupied spaces useful for robot exploration and high-speed navigation.

3.3.1 Problem Formulation

The goal of the network architecture is to learn a function that maps an input occupancy map to an expanded occupancy map that extends beyond the FOV of the sensor. More formally, the following function is learned:

$$f : m_i \to m_p$$

where m_i represents the state, in this case, the input occupancy map as an image, m_p represents the predicted, expanded output occupancy map. Components of the function f include an encoding function $f_{enc}(m_i) \to h \in \mathcal{H}$ which maps the state space, input occupancy maps to a hidden state and $f_{dec}(h) \to$

(m_p), which is a decoding function mapping the hidden state to an expanded, predicted occupancy map.

3.3.2 Neural Network Architecture

Because the generated, predicted occupancy map contains highly correlated data from the input image, I selected an autoencoder architecture based on U-Net [40, 52]. The modified U-Net architecture consists of skip connections which allows a direct connection between layers i and $n-i$ enabling the option to bypass the bottleneck associated with the downsampling layers in order to perform an identity operation. The encoder network consists of 8 convolution, batch normalization and ReLU layers where each convolution consists of a 4×4 filter and stride length of 2. The number of filters for the 8 layers in the encoder network are: (64, 128, 256, 512, 512, 512, 512, 512). The decoder network consists of 8 upsampling layers with the following number of filters: (512, 1024, 1024, 1024, 1024, 512, 256, 128).

3.3.3 Loss Functions

In this section, I describe the loss functions used to minimize the error between the predicted occupancy map, $\hat{m}_p$ and the ground truth image, m_{gt}. The goal was to encourage image reconstruction, while maintaining structural similarity, reducing output noise and emphasizing physical boundary conditions. As described by Equation 3.1, to encourage image reconstruction, I use an l1 loss over l2 based on recent results by [53] demonstrating a reduced blurring effect.

$$\mathcal{L}_{l1}(m_{gt}, \hat{m}_p) = \frac{1}{N} \sum_{i,j} |\hat{m}_p^{i,j} - m_{gt}^{i,j}| \tag{3.1}$$

where N is the number of pixels in the image and i and j represent image pixels. Equations 3.2 and 3.3 describe the second term of the loss function used to maintain structural similarity and measure perceived changes between generated and ground truth images.

$$\mathcal{L}_{ssim}(\hat{m}_p, m_{gt}) = 1 - SSIM(\hat{m}_p, m_{gt}) \tag{3.2}$$

$$SSIM(\hat{m}_p, m_{gt}) = \frac{(2\mu_{\hat{m}_p}\mu_{m_{gt}} + C_1) + (2\sigma_{\hat{m}_p m_{gt}} + C_2)}{(\mu^2_{\hat{m}_p} + \mu^2_{m_{gt}} + C_1)(\sigma^2_{\hat{m}_p} + \sigma^2_{m_{gt}} + C_2)} \tag{3.3}$$

In order to reduce the amount of noise in the generated occupancy map, I apply a total variation loss which has been shown to reduce output noise while preserving edges in the generated image.

$$\mathcal{L}_{tv}(\hat{m}_p) = \sum_{i,j} \left((\hat{m}_p^{i,j+1} - \hat{m}_p^{i,j})^2 + (\hat{m}_p^{i+1,j)} - \hat{m}_p^{i,j})^2 \right)^{\frac{1}{2}} \tag{3.4}$$

In generating predictions of future occupancy maps, generating accurate predictions of the physical boundaries is paramount for collision-free planning. For this reason, I added a fourth term to the loss function that computes an l1 loss on the image gradients with the goal of highlighting edges corresponding to physical boundaries.

$$\mathcal{L}_{ig}(\hat{m}_p, m_{gt}) = \sum_{i,j} \big| |\hat{m}_p^{i,j} - \hat{m}_p^{i-1,j}| - |m_{gt}^{i,j} - m_{gt}^{i-1,j}| \big| + \big| |\hat{m}_p^{i,j-1} - \hat{m}_p^{i,j}| - |m_{gt}^{i,j-1} - m_{gt}^{i,j}| \big| \tag{3.5}$$

The four loss terms are combined as a linear combination where λ_{l1}, λ_{ssim}, λ_{tv}, and λ_{ig} are weights to trade-off the effects of each component of the loss function as described by Equation 3.6.

$$\mathcal{L}(\hat{m}_p, m_{gt}) = \lambda_{l1}\mathcal{L}_{l1}(\hat{m}_p, m_{gt}) + \lambda_{ssim}\mathcal{L}_{ssim}(\hat{m}_p, m_{gt}) + \lambda_{tv}\mathcal{L}_{tv}(\hat{m}_p) + \lambda_{ig}\mathcal{L}_{ig}(\hat{m}_p, m_{gt}) \tag{3.6}$$

3.3.4 Baseline Implementations

I compare this method to several baseline implementations that have successfully been applied to inpainting applications.

(A) a generative network based on a U-Net architecture using only L1 loss(**unet_l1**)

(B) a generative network based on the ResNet architecture (**resnet**)

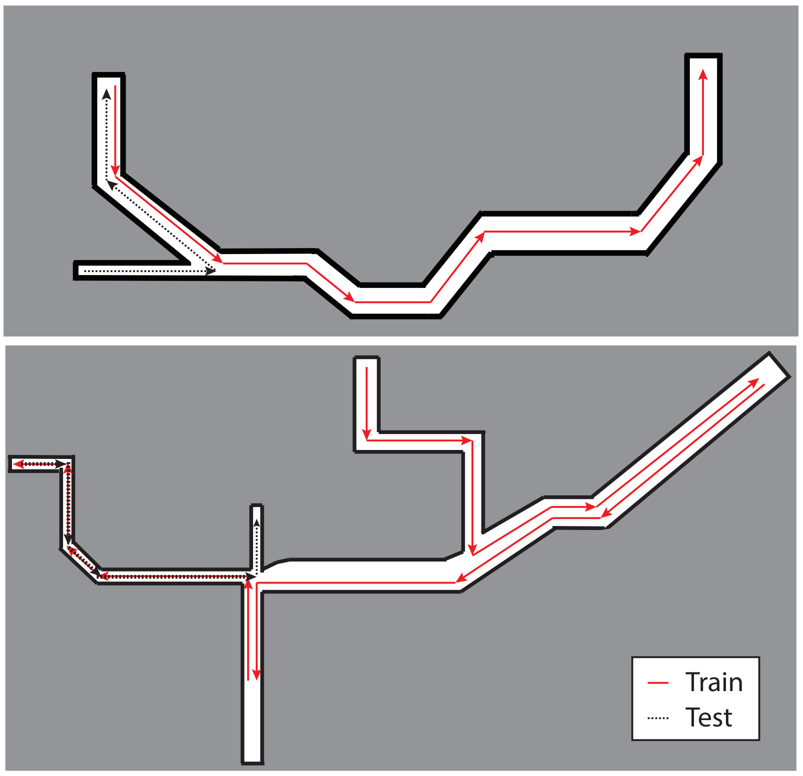


Figure 3.3: Simulated ground truth maps with white representing free space, black is occupied, and gray is unknown. Four trajectories were used for training (depicted in solid red), and two paths were used as the test set (dotted black).

(C) a GAN using the network from (a) as the generative network (**gan**)

3.3.4.1 U-Net Generative Model with L1 Loss

The U-Net feedforward model is based on the network architecture defined by Ronneberger et. al [52] as described above. In the baseline experiments, I use the same U-Net network architecture with skip connections however limiting it to the L1 image reconstruction loss.

3.3.4.2 ResNet Feedforward Model

The ResNet feedforward model is based on the work by Johnson et. al [54] which consists of 2 convolution layers with stride 2, 9 residual blocks as defined by [55] and two deconvolution layers with with a stride of $\frac{1}{2}$. A key reason this network was selected was based on the ability to learn identify functions, which is key to image translation as well as the success in image-to-image translation demonstrated by the CycleGAN network [56].

3.3.4.3 GAN Model

The GAN networks is based on the pix2pix architecture [40] which has demonstrated impressive results in general purpose image translation including generating street scenes, building facades and aerial images to maps. This network uses the U-Net Feedforward model defined in section 3.3.4.1 and consists of a 6 layer discriminator network with filter sizes: (64, 128, 256, 512, 512, 512).

3.3.5 Simulation Experiments

The approach to testing occupancy map prediction using the networks defined above first involved generating a dataset and then performing qualitative and quantitative analysis of the predicted images compared to the ground truth.

3.3.5.1 Data Collection

A dataset of approximately 6000 images of occupancy map subsets was created by simulating a non-holonomic robot moving through a two-dimensional map with a planar LIDAR sensor in C++ with ROS and the OctoMap library [57]. Two maps, shown in Fig. 3.3, were created in Solidworks with the path width varying between 3.5 m to 10 m. These were converted into OctoMap's binary tree format using binvox [58, 59] followed by OctoMap's binvox2bt tool. The result is an occupancy map with all unoccupied space set as free. I require space outside of the walls, shown as grey in Fig. 3.3, to be marked as unknown to provide a ground truth for the estimated maps. These ground truth maps were created by fully exploring the original occupancy maps.

The robot is modeled as a Dubin's car, with a state vector $\mathbf{x} = [x, y, \theta]$ and inputs $\mathbf{u} = [v, \dot{\theta}]$ where (x, y) is the robot's position, v is the velocity, and θ and $\dot{\theta}$ are the heading angle and angular velocity, respectively. For simplicity, the robot is constrained to move at fixed forward velocity of 0.5 m/s. A planar LIDAR sensor with a scanning area of 270° and range of 20 m is used to simulate returns given the robot's current pose against the ground truth map. These simulated returns are used to create the "estimated" occupancy map. Path planning is done with nonlinear model-predictive control and direct transcription at 10 Hz. At each time step, a subset of the maps (both the estimated and ground truth) are saved. A 5 m by 5 m square centered around the robot's pose was chosen with a resolution of 0.05 m. At each time step, the robot's current state and action space are also logged. Occupancy maps are expanded over time, so the simulation performs a continuous trajectory and the data set is built consecutively instead of randomly sampling throughout a map. A total of six trajectories were simulated. Four paths were used for training data (5221 images) and two were used as a test set (1090 images). Ground truth datasets of the expanded occupancy maps were also generated. These expanded occupancy maps range from 1.10x to 2.00x expansion in increments of 0.10x, e.g., a 2.00x expansion results in a 10 m by 10 m square subset centered around the robot.

3.3.5.2 Training Details

I trained each variant of the neural network using the expanded ground truth occupancy maps from scratch for 200 epochs with a batch size of 1. A total of 15 training sessions were performed to evaluate each of the three neural network architectures across five expansion increases (1.10x, 1.30x, 1.50x, 1.70x, and 2.00x). I use the Adam optimizer with an initial learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$. In the feedforward models, L1 loss was used as proposed in PatchGan [40] and in the GAN model

L1+discriminator loss was used. The decoder layers of the network used a dropout rate of 0.50 and weights were initialized from a Normal distribution ($\mu = 0, \sigma = 0.2$). In the experiments, I set $\lambda_{l1} = 100.0$, $\lambda_{ssim} = 10.0$, $\lambda_{tv} = 1.0$, $\lambda_{ig} = 0.1$. These were empirically determined to ensure even contribution of each loss term to the overall loss. All models were implemented using PyTorch [60].

3.3.5.3 Simulation Results

I evaluated the performance of each neural network architecture across a span of five increasing occupancy map predictions. Fig. 3.4 provides a snapshot of the qualitative assessment of the predicted images for each of the neural networks. This example was selected because it demonstrates that even with very little information, this model as well as the U-Net with L1 loss were able to accurately predict the presence of the surrounding obstacles while the other networks were unable to detect it. For a more quantitative comparison, Table 3.1 provides the structural similarity index metric (SSIM) for each of the networks. Based on the SSIM metric, it can be seen that this method outperforms the other approaches for all five tested expansions (1.10x - 2.0x). As expected, the quality of the prediction generally decreases as the expansion percentage increases however this approach generally maintains good performance.

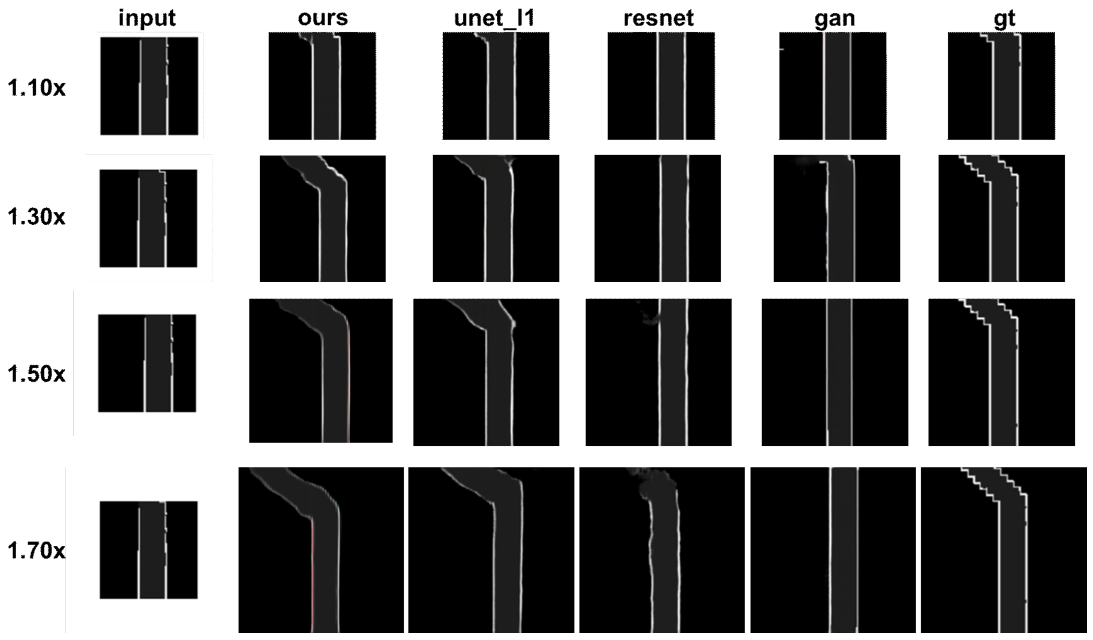


Figure 3.4: This figure describes the input data, the predicted images and the ground truth for each of the neural networks evaluated on the simulated dataset across an expanding prediction window from 1.10x increase to 1.70x increase.

Table 3.1: SSIM Analysis for Simulation Data

Method	**1.10x**	**1.30x**	**1.50x**	**1.70x**	**2.00x**
U-Net (L1) [52]	0.908	0.839	0.780	0.794	0.795
ResNet [54]	0.879	0.814	0.808	0.786	0.800
GAN [40]	0.846	0.825	0.794	0.770	0.641
Our method	**0.926**	**0.866**	**0.846**	**0.825**	**0.818**

3.3.6 Physical Experiments

The next experiment focused on validating this approach with occupancy maps generated by a physical LIDAR sensor. In this experiment, I teleoperated a TurtleBot2 robot with a mounted Hokuyo UST-20LX LIDAR sensor (shown in Fig. 3.1(a)) inside a building. The OctoMap library [57] along with a custom C++ implementation of a particle filter running at 20 Hz was used for simultaneous localization and mapping. Note I use a planar robot, so a 2D slice of the final 3D occupancy map which corresponds to the height of the LIDAR sensor is used for the ground truth (shown in Fig. 3.1(b)). At each time step a 5 m by 5 m square subset centered around the robot's current pose of both the ground truth and estimated maps was saved (100 images). Expanded ground truth occupancy maps were generated ranging from 1.10x to 2.00x in 0.10x increments.

The objective was to evaluate whether training performed on a simulated dataset could be directly transferred to occupancy maps generated by a physical LIDAR sensor. For this reason, I opted to not fine tune the networks using the physical dataset.

Table 3.2: SSIM Analysis for Physical Data

Method	1.10x	1.30x	1.50x	1.70x	2.00x
U-Net (L1) [52]	0.580	0.594	0.567	0.545	0.533
ResNet [54]	0.597	0.605	0.554	0.563	0.549
GAN [40]	0.611	0.602	0.583	0.546	0.493
Our method	**0.630**	**0.637**	**0.626**	**0.600**	**0.577**

3.3.6.1 Physical Experiment Results

Fig. 3.5 represents sample predictions obtained by running the networks trained using simulation data on the occupancy maps generated by the physical sensor. Table 3.2 displays the SSIM metric across each of the networks. In the physical experiments, this method again outperforms the other baseline implementations.

3.3.6.2 Ablation Study

The final study was to conduct a series of experiments to measure the contribution of each component of the four term loss function. The SSIM results for 1.10x expansion using both the simulated and physical test sets are presented in Table. 3.3. As the results indicate, a large improvement is made by adding the SSIM and TV losses. Adding the image gradient loss had negligible impact on the simulation data but did have a greater impact on the physical data. One hypothesis for this observation is that the physical data contains more physical boundaries, so therefore, it would see a greater effect from the

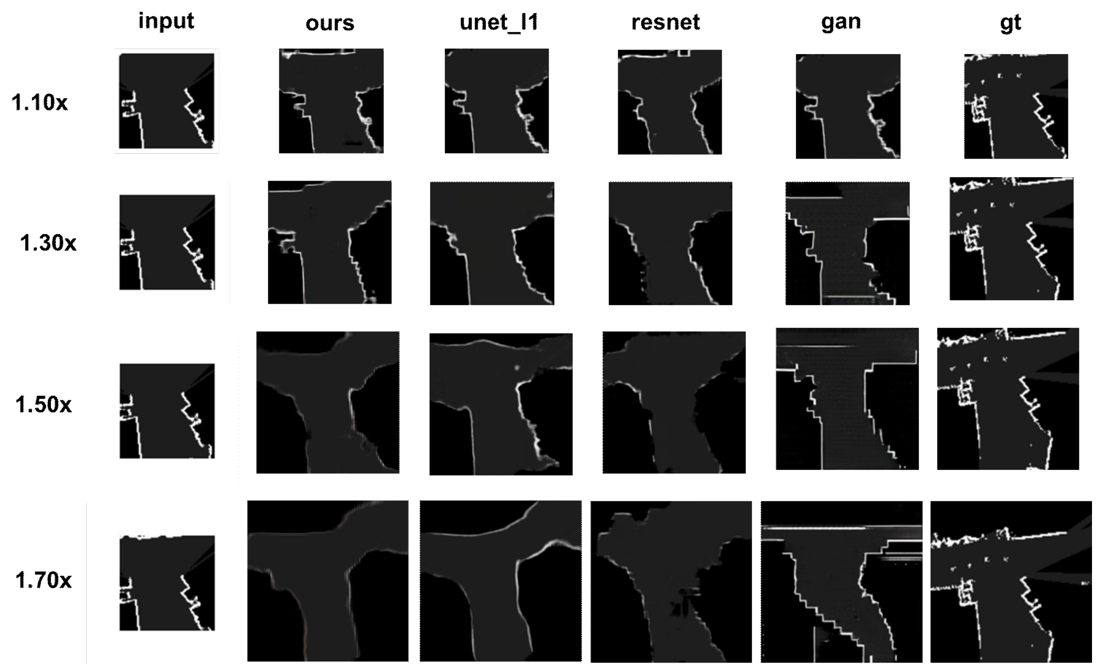


Figure 3.5: This figure describes the input data, the predicted images and the ground truth for each of the neural networks evaluated on the physical dataset across an expanding prediction window from 1.10x increase to 1.70x increase.

image gradient loss.

Table 3.3: Ablation Study Results

Method	**l1**	**l1 + ssim**	**l1 + ssim + tv**	**l1 + ssim + tv + ig**
Our method (sim)	0.908	0.917	0.926	0.926
Our method (phys)	0.580	0.622	0.625	0.630

3.3.6.3 Evaluation on Public Datasets

I further evaluated this approach using publicly available mapping datasets provided by Google Cartographer [16]. The website corresponding to this project contains a repository of numerous Robot Operating System (ROS) bag files from different robotic and non-robotic platforms including a backpack mounted lidar, a PR2 robot from Willow Garage, Turtlebot and the Magazino Robot. For this dataset, I used the *2011-08-03-20-03-22*, *2011-08-04-12-16-23*, *b0-2014-10-07-12-43-25* and the *cartographer_turtlebot_demo* bag files for training and *2011-08-04-23-46-28* and *b2-2015-08-18-11-42-31* bag files for test. The bag files were manually split to ensure non-overlapping training and test data. These bag files consist of laser scan, images, and point cloud data of robots navigating a variety of indoor building environments.

To construct this dataset, I leverage the Google Cartographer ROS package to first build the full 2D occupancy map by playing the entire ROS bag to completion. I save the full map so that it is available as ground truth. I then replay the ROS bag file, this time collecting a pair of images that represents a 4x4 m square with the robot in the center and an expanded occupancy map of size 6x6 m to represent a 1.5x expanded prediction or 1 m in every direction. These pairs of images are then used to train the neural network and validate this approach. In total, I collected a dataset of 4997 training image pairs and 1368 test image pairs.

I trained each variant of the neural network using the expanded ground truth occupancy maps from scratch for 50 epochs with a batch size of 4. I use the Adam optimizer with an initial learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$. The decoder layers of the network used a dropout rate of 0.50 and weights were initialized from a Normal distribution ($\mu = 0, \sigma = 0.2$). In these experiments, for the loss function, I set $\lambda_{l1} = 100.0$, $\lambda_{ig} = 0.1$, $\lambda_{ssim} = 10.0$, $\lambda_{tv} = 1.0$. All models were implemented using PyTorch [60].

I evaluated the performance of the occupancy map prediction and compared it to the baseline implementations described above using both quantitative and qualitative methods. The evaluation metrics used for quantitative comparison are structural similarity metric (SSIM) and peak signal-to-noise (PSNR) between the predicted regions of the image and the ground truth. The PSNR metric provides per pixel level accuracies while the SSIM metric focuses more on perceived changes in the image. These metrics are widely used to compare the relative difference between images and have been applied to various image completion studies [40, 61]. I also evaluate the prediction speed of each approach as an additional performance metric. Each of the algorithms was executed on a laptop consisting of an Intel® i7-7700HQ 2.80 GHz x 8 processor and a NVIDIA® GeForce® GTX 1080 (Max-Q) 8 GB graphics card.

A summary of the quantitative results is presented in Table 4.5 and sample images providing qualitative comparisons are presented in Fig. 3.6. In these results, I show that qualitatively, this approach generates better prediction results compared to the baseline experiments. Quantitative assessments show that this approach generated the maximum PSNR score however performed

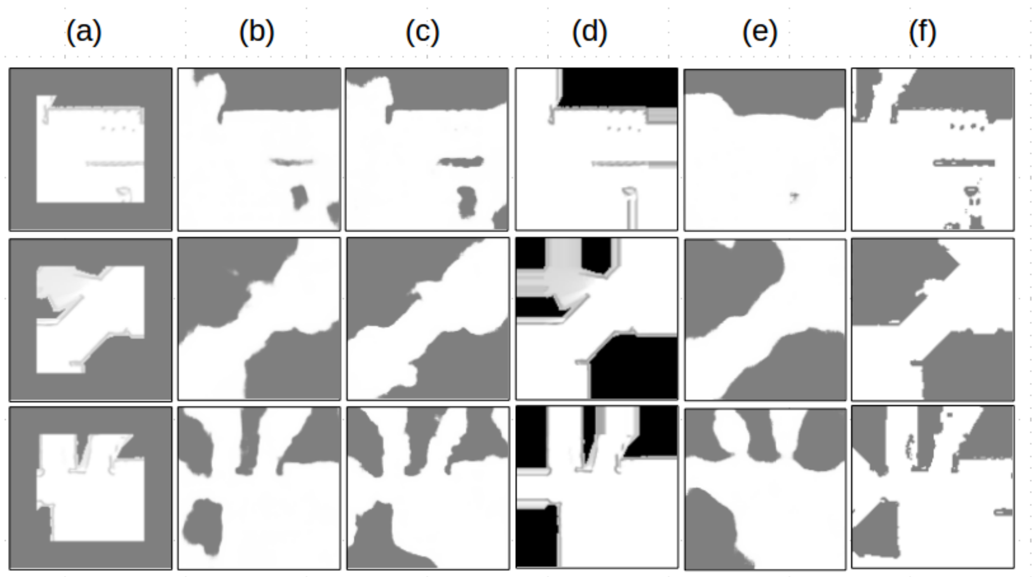


Figure 3.6: (a) Input occupancy map based on lidar's FOV, (b) Prediction using our method, (c) Prediction using U-Net with L1 loss, (d) Prediction using border padding algorithm, (e) Autoencoder network, and (f) Ground truth image

Table 3.4: Quantitative Analysis for Predicted Occupancy Map

Method	**SSIM**	**PSNR**	**Speed (ms)**
Border Completion	0.792	9.731	0.2
Autoencoder Network (w/o skips)	0.904	16.459	4.2
U-Net (L1) [52]	0.896	16.726	4.3
Our Method	0.903	16.907	4.3

slightly worse than the simple autoencoder network when comparing the structural similarity. One potential explanation is that the SSIM metric tends to minimize small deviations by searching for neighboring pixels whereas the PSNR score will capture these small deviations.

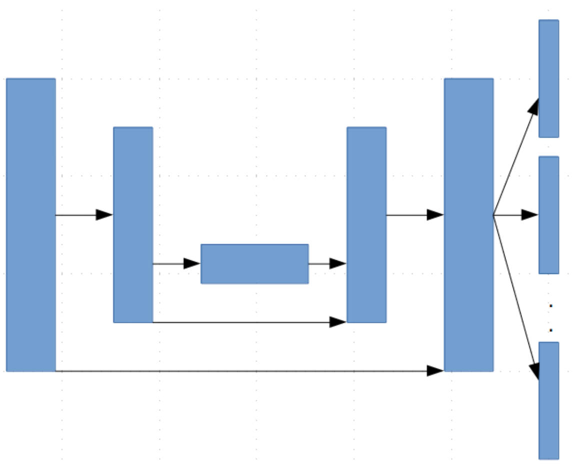


Figure 3.7: U-Net architecture extended to generate multiple hypotheses

3.4 Information-Theoretic Exploration

The goal in this section is to leverage map prediction to improve robot exploration strategies when mapping a new environment. I extend the prior map prediction algorithm to include uncertainty estimation and use the uncertainty as an information-theoretic approach to exploration.

3.4.1 Uncertainty-Aware Prediction

While the approach described above using a single hypothesis prediction produces reasonable performance, the reality is that the world is inherently ambiguous and uncertain, particularly when navigating environments. To illustrate this, consider the example of a robot approaching the start of a hallway. At this point, there is inherent uncertainty in the future prediction as the hallway could result in a T-intersection or lead to an exit.

In this scenario, making a prediction using a single hypothesis will often result in a blurry image consisting of the two possible futures as depicted by Fig. 3.8 (b). A better approach would be to generate multiple predictions that capture the distribution of possible futures and plan along the different hypotheses. As more information is obtained, the goal is that the hypotheses will converge to a single prediction.

3.4.2 Generating Multiple Hypotheses

To capture this uncertainty, I modify the single hypothesis network to output multiple hypotheses predictions as described by Rupprecht et al. [62]. I do this by branching N heads with each head capable of making its own prediction as summarized in Fig. 3.7. The loss function is modified to become a weighted sum, $1 - \epsilon$, of the best performing head loss and the weighted sum, $\epsilon/(N-1)$, of the losses of the other heads. In these experiments, I set $\epsilon = 0.05$.

3.4.3 Multiple Hypotheses Prediction Results

In this section I present qualitative and quantitative results representing multiple hypothesis prediction.

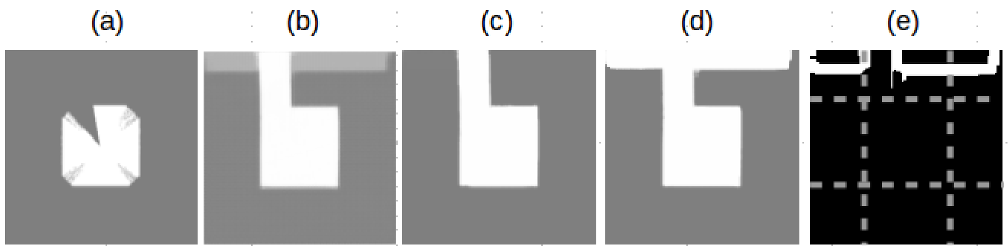


Figure 3.8: (a) Input occupancy map based on lidar's FOV, (b) Prediction using single hypothesis resulting in blurred image, (c) and (d) 2 hypotheses generated using multiple hypotheses prediction, (e) Image representing variance between the multiple hypotheses

3.4.3.1 Qualitative Results

In Fig. 3.8 (c) and (d) where $N = 2$, I display the result of making multiple predictions to generates hypotheses. In this figure, we can clearly see two distinct hypotheses that represent that true uncertainty of the environment rather than a blurred version corresponding to the combination of the possible environments in Fig. 3.8 (b).

3.4.3.2 Quantitative Results

I then show the effects of making multiple hypotheses on a more typical indoor navigation dataset. I consider the same test datasets described in Section 3.3.6.3 and compare the SSIM and PSNR metrics when generating 4 and 8 hypotheses to the best results obtained with a single hypothesis. I first compare the best of N hypotheses referred to as MHP-N (best) in Table 3.5. I accomplish this by taking the highest SSIM and PSNR scores of the N predictions and observed and compared to the best results using a single hypothesis. I further compared the average of N hypotheses and found that even the average of the hypotheses outperforms the best single hypothesis showing that making multiple predictions is making an impact on generating better overall predictions. An explanation for this is that while there is ambiguity in the predictions, as more information is obtained, the multiple hypotheses eventually converge to a better prediction.

Table 3.5: Quantitative Analysis with multiple hypotheses prediction

Method	**SSIM**	**PSNR**	**Speed (ms)**
Our Method - Single Hypothesis	0.903	16.907	4.3
MHP-4 (avg)	0.911	17.204	16.1
MHP-8 (avg)	0.912	17.157	32.3
MHP-4 (best)	0.919	18.022	16.1
MHP-8 (best)	0.921	18.252	32.3

3.4.4 Efficient Map Exploration

3.4.4.1 Algorithm

I now consider an approach to leverage the uncertain regions of the multiple hypotheses as a heuristic for an information-theoretic map exploration strategy as described by Algorithm 1. To accomplish this, I generate a variance image from the N hypotheses that describe regions of the hypotheses with greatest dissimilarities as shown in Fig. 3.8 (e). In this work, to generate the variance image, I use the SSIM image difference across the N hypotheses rather than raw pixel differences to make the variance image more resilient to minor pixel changes. I further threshold the image using both a binary and Otsu's Method [63] for noise reduction. I then partition the variance image into M clusters. In these experiments, I apply a simple 3x3 grid on the variance image (Fig. 3.8 (e)) and determine the grid containing the maximum variance. Finally, I find the centroid of this grid and use this to compute a new robot pose for exploration.

Algorithm 1 Map Exploration Algorithm

```
    Input $N$ hypotheses, $M$ clusters, $B$ bounding box to explore
    Output $Pose$ next pose of the robot
1: procedure ExploreMap($N, M$)
2:     while $B$ Not Fully Explored do
3:         $H[N] \leftarrow GenerateHypotheses(N)$
4:         $V \leftarrow GenerateVarianceImage$
5:         $Cluster[M] \leftarrow ClusterVarianceImage(V, M)$
6:         $MaxCluster \leftarrow MAX(Cluster[M])$
7:         $Pose \leftarrow Centroid(MaxCluster)$
8:         $MoveRobot(Pose)$
```

3.4.4.2 Simulation Platform

I test this algorithm using the Gazebo simulation environment consisting of a Turtlebot with an onboard lidar. I compare this exploration strategy with frontier exploration [3], a modified version of frontier exploration that takes into account the distance to the next frontier [6] and an information theoretic approach using Bayesian optimization [4,5].

3.4.4.3 Simulation Results

Fig. 3.9 shows the trajectory of the robot using each of the baseline algorithms compared to this approach with the predicted regions overlayed. The total lengths of the trajectories are also presented in Table 3.6. The main observation here is the decision made by the other exploration algorithms to explore the corner of the map where very little information can be gained versus the exploration strategy which seeks the hallway where more information can be obtained.

Table 3.6: Total Path Length

Method	**Path Length**
Frontier Exploration [3]	28.17 m
Information-Theoretic Bayesian Optimization [4,5]	17.30 m
Frontier Exploration using Distance [6]	35.26 m
Our Method	**10.05 m**

3.5 High-Speed Navigation

In the previous section, I demonstrated how occupancy map prediction can improve the performance of exploring a new environment. In this section, the primary goal is to develop a controller that leverages the predicted regions of the map to achieve high speed navigation. I execute the tests on an RC car as

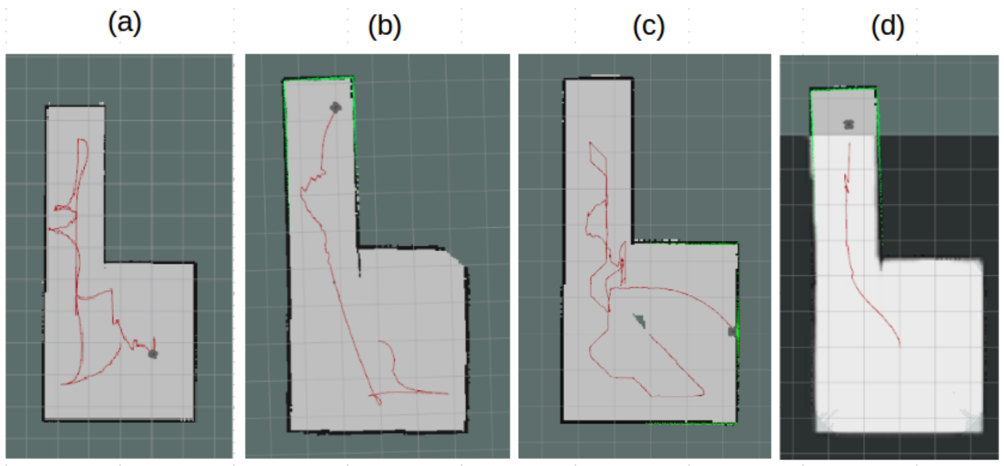


Figure 3.9: Trajectory of robot during exploration using (a) Frontier Exploration [3], (b) Information-Theoretic Bayesian Optimization [4,5], (c) Frontier Exploration using Distance [6] , (d) Our Method

described in Sec. 3.5.1 and make use of RGBD mapping, map prediction, and a receding-horizon controller to achieve this goal.

3.5.1 Platform

The platform I use for the evaluation is the MIT Race car [64] built on the 1/10-scale Traxxas Rally Car platform, as shown in Fig. 3.11. This RC car has a reported maximum speed of 40 m/s and contains Intel Realsense D435 and T265 cameras. The onboard Nvidia® Jetson TX2 computer runs the perception, mapping and planning software integrated with the Robot Operating System (ROS) [65]. The main interface to the RC car is the variable electronic speed controller (VESC) interface that provides vehicle state information and receives commands including desired velocity and turn angle.

3.5.2 Approach

A summary of the approach is described in Fig. 3.10 which consists of perception and control blocks.

3.5.2.1 Perception

The objective of the perception algorithm is to observe co-registered RGB and depth data to produce an occupancy map for planning. As demonstrated in Fig. 3.10, RTAB-Map [17] is used to create 2D occupancy maps using the RGB and depth images from a Realsense D435 and visual inertial odometry from a Realsense T265.

To improve the mapping performance, I limit the D435's depth sensor range to 3 meters, and I apply gradient filtering on the raw depth images. The depth image gradients are calculated using a Sobel filter with a 5×5 kernel. All pixels with a gradient magnitude larger than twice the median are discarded. This removes "ghost noise" near sharp edges in the image. I use RTAB-Map to generate an updated map at approximately 3 Hz while running on the Nvidia® Jetson TX2 hardware on the car.

3.5.2.2 Neural Network Architecture

I use a U-Net style neural network architecture [40, 52] as described in Sec. 3.3 to receive the occupancy map provided by RTAB-Map and predict

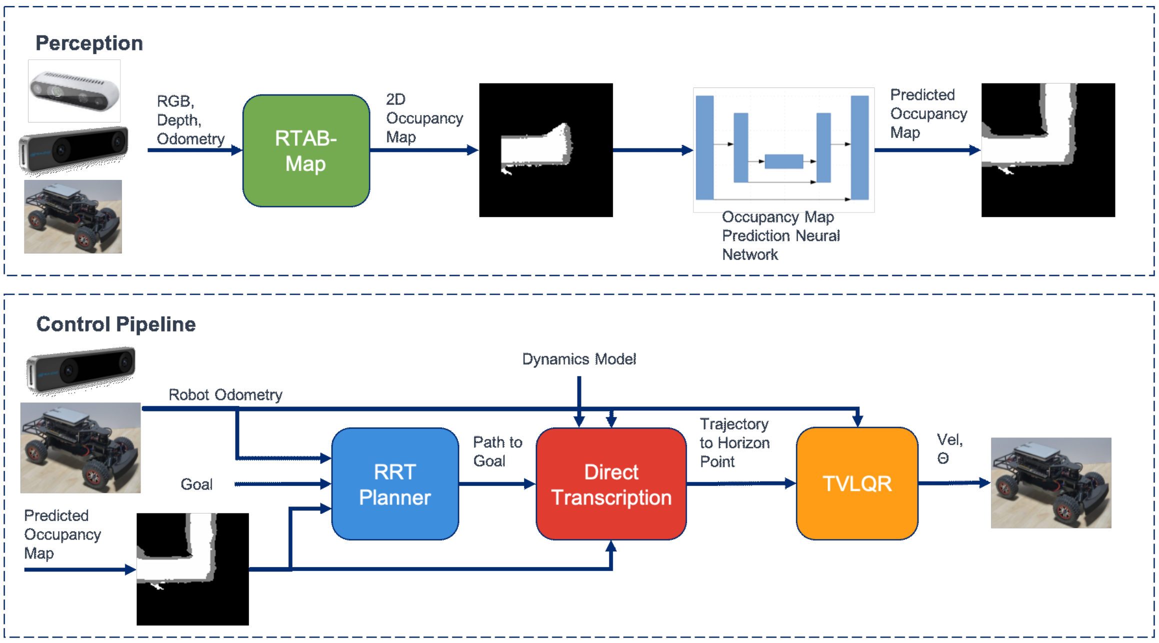


Figure 3.10: This diagram describes the overall perception and control pipeline. The perception module receives on-board sensor data from the car and produces a predicted occupancy map using a U-Net style generative neural network. The control algorithm receives the robot state, predicted occupancy map and goal point and generates collision-free trajectories.

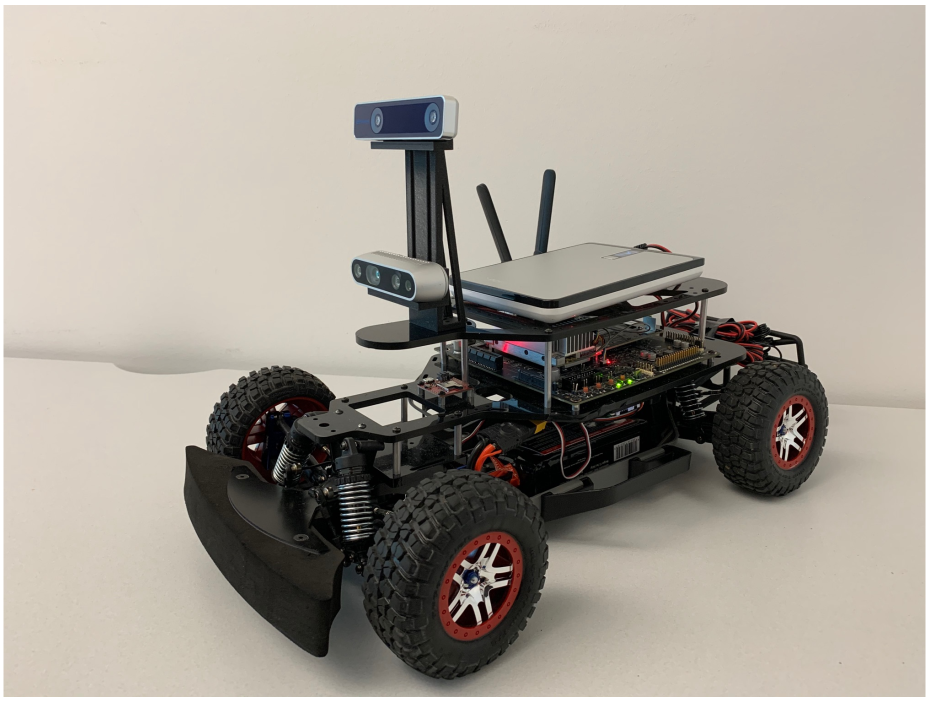


Figure 3.11: MIT Racecar with Intel Realsense Cameras

an expanded occupancy map. In summary, the U-Net neural network is a generative architecture used in several image completion algorithms including [40, 66, 67]. The U-Net network consists of skip connections allowing a direct connection between the layers i and $n-i$. These skip connections enable the option to bypass the bottleneck associated with the downsampling layers and significantly increases the accuracy of predicted occupancy regions [68]. In this work, the implementation of the encoder network consists of 7 convolution, batch normalization and ReLU layers where each convolution consists of a 4×4 filter and stride length of 2. The number of filters for the 7 layers in the encoder network are: (64, 128, 256, 512, 512, 512, 512). Similarly, the decoder network consists of 7 upsampling layers with the following number of filters:

(512, 1024, 1024, 1024, 512, 256, 128).

3.5.2.3 Loss Function

To train the network, I extend the loss function described in Sec 3.3, by using a class-balanced cross-entropy loss function as described in [69]. The discrete classes used for labeling each pixel of the occupancy map include occupied, unoccupied and unknown spaces. Because there are significantly more pixels associated with unoccupied and unknown spaces versus obstacles, I apply class balancing techniques on the cross entropy loss with additional $5\times$ weight added to the occupied space loss. This results in predictions where the edges representing obstacles are far more pronounced as seen in Fig. 3.12.

3.5.2.4 Post Processing

During the testing on the robotic car, I frequently observed small noise artifacts being generated by the neural network, a condition commonly found in generative neural networks [70]. While seemingly minor and transient, these artifacts caused significant instability issues during control as the trajectory planner would often abruptly change the planned path in response to these artificial obstacles or would fail to find a valid trajectory. To alleviate this, I apply a traditional morphological closing operation with a 5×5 kernel to suppress the noise generated by the neural network with results shown in Fig. 3.13. The observed map and the filtered predictive map are combined to create the planning map; any unknown space from the observed map is filled in with data from the predictive map.

3.5.2.5 Training Details

I generate the datasets in an unsupervised manner. As the robot navigates a new environment, the robot collects data that consists of a submap that corresponds to the current occupancy map based on the sensor's horizon as well as the expanded ground truth map after the environment has been explored. I explored various sizes of the submap and found a map corresponding to 6 m $\times$ 6 m provided by best geometric size of the submap given the characteristics of the sensor. I used a map resolution of 0.05 meters per pixel so the input occupancy map image resolution was 120 $\times$ 120 pixels. Further, I experimented with various predicted region sizes and found predicting a region of 7.5 m $\times$ 7.5 m corresponding to an image size of 150 $\times$ 150 provided the optimal accuracy and performance characteristics for the controller. Further, due to limited amounts of real world data available, I perform data augmentation techniques to apply random rotations to the occupancy map training data. This allows us to be more robust to various hallway configurations as shown in Fig. 3.14.

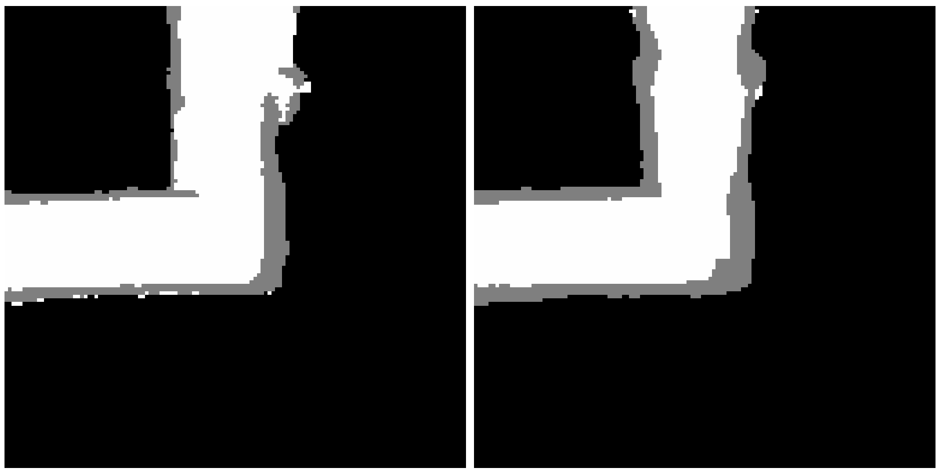


Figure 3.12: Predicted occupancy map generated without class balancing weight (Left) and with class balancing weight (Right) where white represents unoccupied space, grey is occupied and black is unknown. The class balancing weight produces stronger edges for obstacles in the predicted occupancy map.

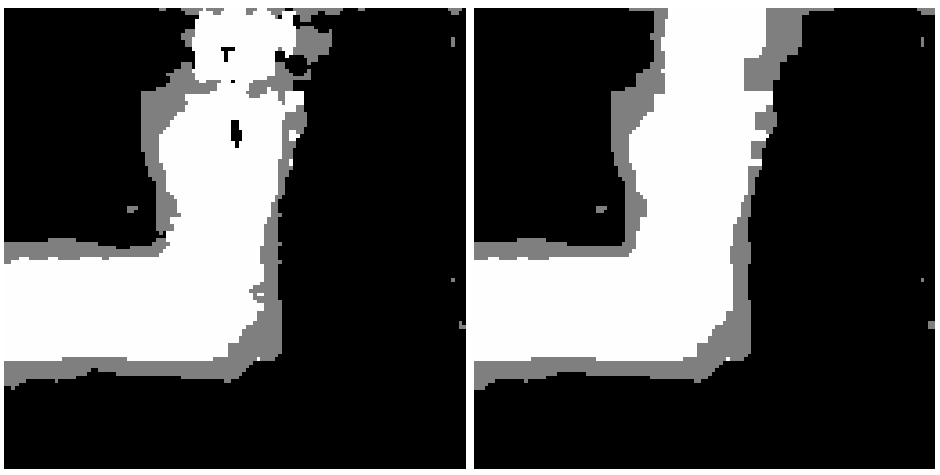


Figure 3.13: (Left) Generated occupancy map with noise artifacts. (Right) Predicted occupancy map after morphological close operation (white is unoccupied space, grey is occupied and black is unknown).

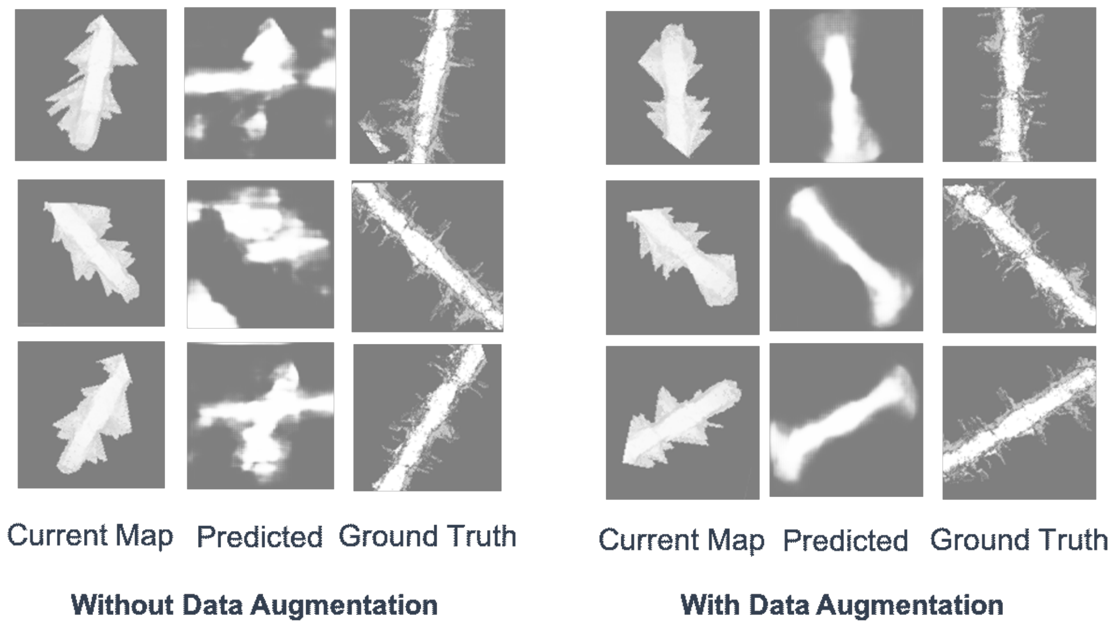


Figure 3.14: Three examples from the training set of occupancy maps and their resulting expanded predictive map along with the ground truth (white is unoccupied space, light grey is occupied and dark grey unknown). Augmenting the training data with random rotations, allows the network prediction to be more robust to different environment configurations encountered by the robot.

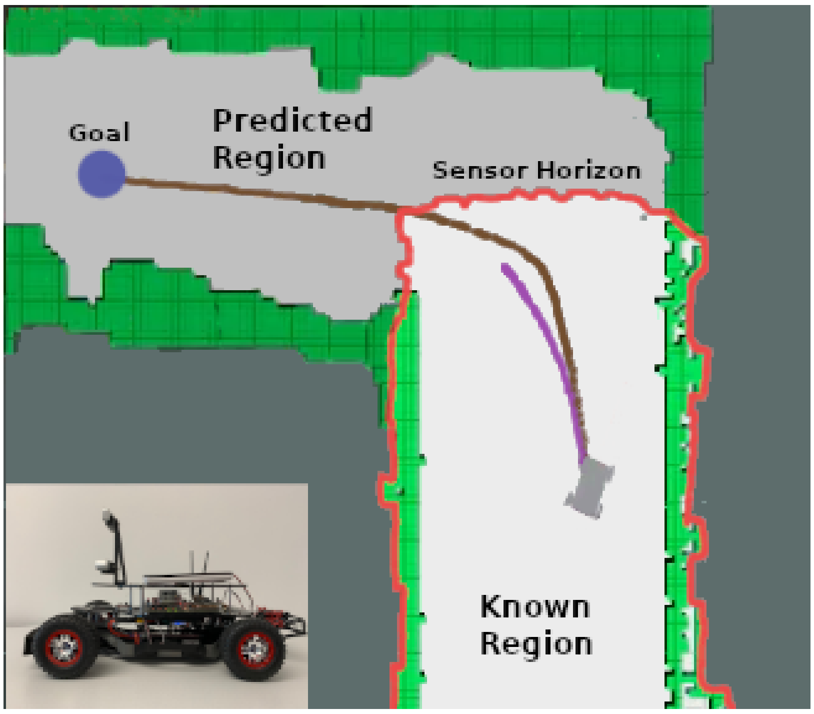


Figure 3.15: Visualization of system during hardware experiment. Known map is enclosed by the red boundary. The brown path is the smoothed RRT path to goal, and the purple path is the local optimized direct transcription trajectory.

3.5.3 Control Algorithm

The objective of the control algorithm as described in Fig. 3.10 is to compute a collision free path to the goal, generate a series of feasible trajectories to waypoints, and send control commands to the mobile robot to follow the computed trajectory. The controller proposed in the prior work [71] was adapted and tuned for this hardware. While initially developed for fixed-wing flight, it is particularly well suited for high-speed navigation. The receding horizon allows for rapid replanning while using a dynamically built map, and the trajectory generation and tracking allows for a high-rate, dynamically feasible control output.

3.5.3.1 Dynamics Model

A simple bicycle acceleration model was used to describe the robot's dynamics. The equations of motion are as follows:

$$
\begin{aligned}
\dot{x} &= v * cos(\theta) \\
\dot{y} &= v * sin(\theta) \\
\dot{v} &= u_0 \\
\dot{\theta} &= v * tan(\delta)/L \\
\dot{\delta} &= u_1
\end{aligned}
\tag{3.7}
$$

The state is written as $x = \begin{bmatrix} x, y, v, \theta, \delta \end{bmatrix}$ where x and y are 2D position, v is forward velocity, θ is orientation, and δ is turn angle. The input, $u = \begin{bmatrix} u_0, u_1 \end{bmatrix}$, represents acceleration and turn angle velocity, respectively, and L is the wheel base length.

3.5.3.2 Control Strategy

Here, I review the receding horizon controller proposed in [71], which can be decomposed into three main stages.

In the first stage, a path to goal is generated using a standard rapidly-exploring random tree (RRT) [72]. The resulting path is pruned and then smoothed using G2 Continuous Cubic Bézier Spiral Path Smoothing (G2CBS)

[73]. The curvature along the smoothed path, $\Big[x(s), y(s)\Big]$, is calculated as:

$$\kappa(s) = \frac{(y''(s)x'(s) - x''(s)y'(s))}{(x'(s)^2 + y'(s)^2)^{\frac{3}{2}}}$$

This curvature is then mapped to velocity based upon v_{max} and v_{min}, the maximum and minimum velocity.

$$v(s) = \frac{dx}{dt}(s) = v_{max} - \kappa(s) * \frac{v_{max} - v_{min}}{2}$$

The path's velocity parameterization is used to reparametrize the path by time.

$$t = \int_0^s \frac{1}{v(s)} ds$$

In the implementation, the RRT was modified to improve performance in a dynamically built map by initializing the RRT tree with the raw RRT path to goal from the previous control iteration. Before initialization, the path was checked for collisions and truncated if a collision is detected. This initialization results in faster RRT computation and increased path consistency between iterations.

In the second stage, a dynamically feasible trajectory from the current state to a horizon point is generated. The horizon point is selected as a time horizon selected along the parameterized RRT path. The same direct transcription feasibility problem was utilized as formulated in [71]. This approach discretizes the trajectory into N knot points using a variable time interval dt. Let ${}_0(t_k)$ be the position of the robot and ${}_0(t_k)$, $k < N$, be the input at the k^{th} knot point where $t_{k+1} = t_k + dt$. This feasibility problem was modified by introducing a cost function to penalize large dt (therefore encouraging high speeds) as well as slightly penalizing the input to encourage smoother trajectories. The objective function is shown below.

$$J({}_0(t_k),{}_0(t_k), dt) = \sum_{k=0}^{N-1} {}_0(t_k)^T \ \mathbf{R_c} \ {}_0(t_k) + dt$$

In order to track the dynamically feasible trajectory, time-varying LQR (TVLQR) is performed in the third stage. The control signal is generated as:

$$(t_k,) = K(t_k)(-{}_0(t_k)) +_0 (t_k).$$

where $K(t_k)$ is the optimal gain matrix.

3.5.3.3 Control Parameters

The control pipeline executes at a rate of 5 Hz, or a control interval of $T = 0.2$ seconds. The control signal is calculated from the odometry and TVLQR gains at a rate of 50 Hz. The max time and max iterations for the RRT search were set to 0.05 seconds and 20000 iterations. The maximum velocity for RRT path parameterization was set to the maximum allowed velocity specified in each hardware trial. For RRT sampling, an obstacle avoidance radius of 0.4 m was used.

For direct transcription, $N = 10$ knot points and a time horizon of $H = 2$ seconds were used. I set $\delta_f = \begin{bmatrix} 0.1, 0.1, 0.1, 0.25, 100.0 \end{bmatrix}$, $\mathbf{R_c} = diag(0.1, 0.1)$ and use an obstacle radius of 0.35 m.

The costs for TVLQR are as follows:

$$Q = diag(10, 10, 10, 10, 10)$$
$$Q_f = diag(1, 1, 5, 1, 1)$$
$$R = diag(1, 1)$$

The acceleration and turn angle control bounds were set to [-2.5, 2.5] m/s and [-1.5, 1.5] rad/s respectively. The velocity minimum bound was set to 0.5 m/s and turn angle state bounds were set to [-0.3, 0.3] rad.

3.6 Experimental Evaluation

I conduct preliminary hardware experiments to validate this approach using a robotic car based on MIT's open-source race car [64] as described in 3.5.1.

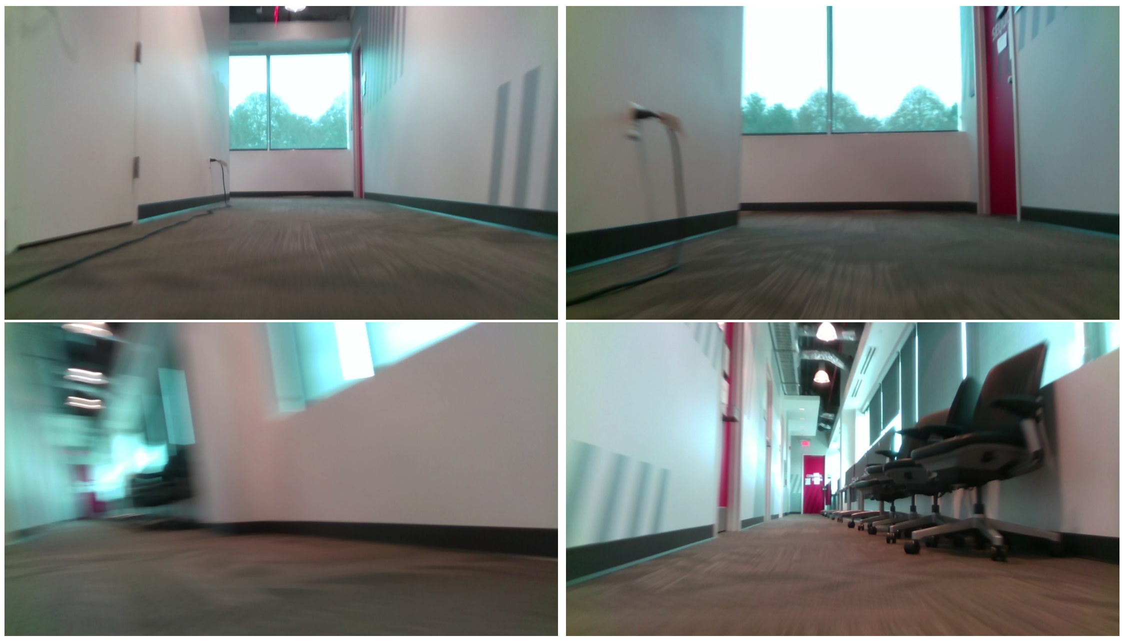


Figure 3.16: This sequence of images represents a trajectory taken by the robot during the experimental evaluation.

The map prediction network was trained on indoor scenes consisting primarily of straight and turning corridors. Similar to [74], I conduct both zero-shot and continual learning scenarios. In the zero-shot experiments, I evaluate the performance on new environments (Fig. 3.16) not seen by the robot. In the continual learning evaluation, I allow the robot to collect data in a semi-supervised manner from the new environment, fine tune the network offline and reevaluate performance. I assessed the maximum speed allowed by the robot and the number of successful trials, which I defined as reaching the target goal without collision. A visualization of the system during the hardware experiment is shown in Fig. 3.15.

Algorithm	**Max Speed**	**Success Rate**
Without Map Prediction	3 m/s	5/5
Without Map Prediction	4 m/s	1/5
With Map Prediction (Zero-shot)	4 m/s	3/5
With Map Prediction (Fine Tuned)	4 m/s	4/5

Table 3.7: This table captures the results of the preliminary hardware experiments on the modified MIT race car.

3.6.0.1 Quantitative Results

The results are summarized in Table 3.7. Without map prediction, the robot's maximum speed, v_{max}, was 3 m/s without collision. When evaluating without map prediction with the maximum speed of 4 m/s, the robot was only able to successfully reach the goal 1/5 attempts. With map prediction, I was able to achieve success 3/5 trials. After allowing the network to fine tune on the new environment, I was able to increase the ratio to 4/5 successful trials showing the ability to continually learn as the robot explores new environments.

3.6.0.2 Qualitative Results

For comparison, the trajectories with and without prediction for one of the trials where $v_{max} = 4\,$m/s is shown in Fig. 3.17. Without map prediction, the robot limited by the sensor's field of view, plans a waypoint in unknown space and is not able to react in time once the map has been updated to reflect the true occupied space. In contrast, with map prediction, the robot is able to plan

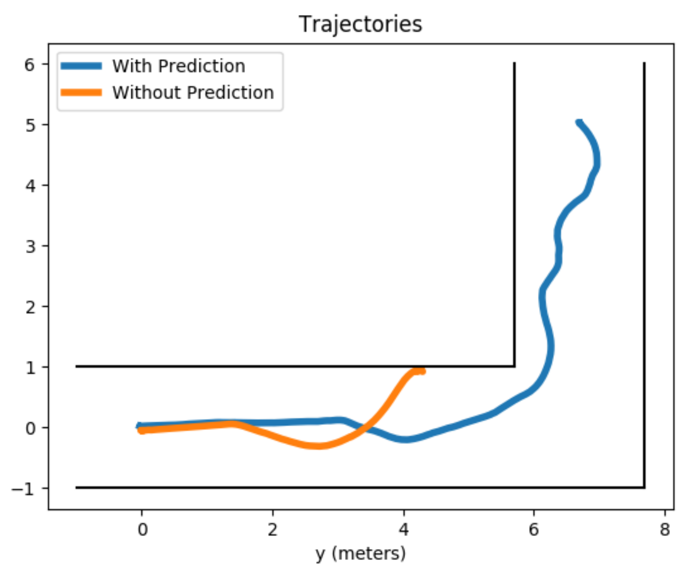


Figure 3.17: Example trajectories of car with max velocity of 4 m/s with and without map prediction.

with longer horizons resulting in smoother trajectories that allow the robot to reach the desired goal.

3.7 Discussion

In this chapter, the goal was to establish a framework to allow robots to predict occupied spaces beyond the line of sight of the sensor and subsequently use the predictions to improve robot exploration and operate at higher speeds. Specifically, the objective is to answer the following four main questions:

1. Can generative neural networks learn a latent representation from occupied maps that act as a predictive model?

2. Can these networks be modified to represent the uncertainty and ambiguity inherent in making predictions?
3. Can knowledge of this uncertainty be used as a strategy for exploring unknown environments?
4. Can map prediction be used as part of a controller to achieve higher speeds?

To answer the first question, I evaluate different neural network architectures and loss functions and find that a U-Net autoencoder architecture with skip connections combined with a multi-term loss function that encourages reconstruction while preserving structure and edges resulted in the best prediction.

To answer the second question, I show that by extending the single hypothesis to multiple hypotheses, I was able to improve prediction performance and capture the uncertainty in both simulated and a more realistic, real world dataset.

To address the third question, I present an algorithm that leverages the differences in the generated hypotheses as a heuristic for efficient exploration. I compare this approach to existing greedy and information-theoretic map exploration techniques and show that this method results in a 41% improvement in trajectory length due to combining elements of prediction with exploration.

Regarding the fourth question, I present a real-time controller that leverages the predicted occupancy map as part of the planning algorithm. At a max velocity of 3 m/s, 3 Hz mapping rate, 3 m sensor range, and 1 sec time horizon, planned trajectories will almost always be within the known region of the map. At these speeds, the system is successful without map prediction with these parameters. When the max velocity is increased to 4 m/s, the planned trajectories will often be within unknown space of the map (outside of the sensor range). This can cause trajectories to plan through unseen walls, causing failure. The predicted occupancy map is able to help address this limitation by providing a longer horizon for planning which accounts for the 4$\times$ improvement at 4 m/s in the preliminary hardware evaluation.

3.8 Conclusion

The long term objective is to develop a full set of capabilities that can take advantage of making predictions for robot planning. In this chapter, I lay the foundation for making predictions based on latent spaces learned from unstructured navigation and navigation policies that leverage the predicted regions of the map. The future work spans multiple activities. I believe the prediction results can be improved by addressing noise issues including scattering effects present in lidar-based sensors. I also plan to focus on more complex environments that consist of additional static and dynamic obstacles. I further plan to investigate new techniques that can better capture the uncertainty of the environment.

While this chapter focused on static obstacles in the environment, an equally important and challenging task is to predict and navigate around dynamic obstacles including pedestrians. In chapter 4, I continue the work by first developing techniques to perform pedestrian state estimation with uncertainty. I follow this by extending the prediction capabilities to include pedestrian prediction using generative networks that combine past trajectories with estimated intent. Finally, I develop an adaptive RL policy that leverages the pedestrian prediction to improve navigation performance.

Chapter 4

Pedestrian Navigation

4.1 Introduction

In the previous chapter, I focused primarily on navigating around static obstacles by leveraging predicted maps. In this chapter, I focus on navigation with dynamic obstacles, in particular pedestrians. As we continue to think about navigation activities ranging from security surveillance to warehouse automation, it is critical for these robots to move efficiently and safely around humans. Navigation around pedestrians is often thought to be a challenging as humans have many unobservable states that govern their own navigation policy, making prediction of future motion challenging. While learning techniques have made tremendous progress in this space, most approaches struggle when operating outside of the training distribution. In this chapter, I focus on first pedestrian state and uncertainty estimation techniques. I show that by combining Kalman filter techniques with deep neural networks, I can generate more robust state and uncertainty estimation at a lower computational cost. I then focus on pedestrian prediction leveraging an estimate of human intent to generate better future predictions evaluated on several pedestrian benchmark datasets. Finally, I develop an RL policy that leverages the predicted motion to produce an adaptive navigation policy and evaluate in both simulation and in physical hardware experiments.

The organization and specific contributions of this chapter are the following:

- In Sec. 4.2, I briefly summarize prior work related to pedestrian state estimation, prediction and navigation.
- In Sec. 4.3, I focus on an approach that combines neural networks and deep Kalman filtering techniques to provide state and uncertainty estimation that is robust to out-of-distribution sensor noise.
- In Sec. 4.4, I extend my neural network prediction research to generate predictions of future pedestrian motion and compare against state-of-the-art methods.
- Finally, in Sec. 4.5, I describe an RL policy that leverages the pedestrian prediction to generate an adaptive navigation algorithm that is more robust to out-of-distribution pedestrian motion.

4.2 Prior Work

State Space Modeling

Classical state estimation has been instrumental in solving a wide variety of problems in the robotics community and can be summarized by several review papers including [75–77]. Recently, many research efforts introduced concepts that combine neural networks with state space models and recursive filters. BackpropKF [78] uses a feedforward neural network to produce a latent embedding and covariance matrix from a raw high dimensional input however uses a known state transition model. This is further extended by DPF [79] which includes the use of particle filters for state estimation with a known dynamics model. Additional works combine variational autoencoders with Kalman Filters including [80, 81]. The work presented in [82] develops a Recurrent Kalman Filter Framework that learns a transition model operating on latent representations with an emphasis factorized inference for efficient computation. DVBF is another work [83] that learns state space models using Bayesian Filters for stable long term predictions. Recently introduced, DynaNet [84] employs state estimation and motion prediction techniques using a neural Kalman Filter and evaluates their approach on visual odometry using the KITTI dataset.

Uncertainty Estimation

Bayesian Neural Networks [85,86] have been used extensively to represent uncertainty in a neural network by learning a probability distribution over the network parameters. Recent works also focus on improving uncertainty estimation using deep neural networks. Kendall and Gal [87] describe techniques that capture aleatoric uncertainty (uncertainty found in observations) and epistemic uncertainty (model uncertainty) particularly in the computer vision domain. Gal and Ghahramani [88] describe uncertainty estimation techniques using stochastic dropout as a form of Bayesian approximation. Ovadia and Fertig [89] evaluate the performance of predictive uncertainty under distributional shift on a variety of modalities including images, text and categorical data.

Monocular 3D State Estimation

Several techniques focus on performing state estimation with a monocular camera. Engel et al. [90] develop large scale SLAM algorithms that operate on a monocular camera with an emphasis on real-time performance. ORB-SLAM [91] extends this work to focus on robustness to motion clutter using loop closure and relocalization. Bertoni et al. develop the Monoloco algorithm [7] which is among the first approaches to perform 3D pedestrian localization from monocular images that captures both aleatoric and epistemic uncertainty, however do so without modeling dynamics.

Crowd Navigation

Previous studies have investigated approaches that enable mobile robot navigation in crowded environments (e.g., [92]). This body of work can be classified into three broad areas: (1) algorithms that react to moving obstacles in real time, (2) trajectory based approaches that plan paths by anticipating future motion of obstacles, and (3) reinforcement learning based approaches that learn a policy to navigate in crowded environments. Reaction based methods include works such as reciprocal velocity obstacles (RVO) [93] and optimal reciprocal collision avoidance (ORCA) [94]. Trajectory based approaches, such as [95, 96], explicitly propagate estimates of future motion over time and perform trajectory optimization on those future states for collision avoidance. Additionally, several recent works use variations of reinforcement learning to learn policies capable of crowd navigation (e.g., [97–99]). Everett et al. [98] developed a decentralized approach to multiagent collision avoidance using a value network that estimates the time to goal for a given state transition. Chen et al. [99] further extended this work by adding an attention mechanism and a novel pooling method to handle a variable number of humans in the scene. Kahn et al. [100] investigates adaptive navigation polices based on uncertainty;

however, they only considered environment uncertainty with static obstacles and not navigation in the presence of pedestrians.

Pedestrian Prediction

Several studies have investigated pedestrian prediction for a variety of applications including robotics, autonomous driving, and video surveillance. Many approaches treat pedestrian prediction as a state estimation problem by relying on a kinematic model and using concepts from Bayesian and Kalman Filtering [101–104]. Many other works have investigated intent or goal based estimation as part of trajectory planning (e.g., [105–107]). Recent works have investigated deep neural networks that consider agent-to-agent and agent-to-environment interactions (e.g., [19–22]). This brief summary only highlights a small snapshot of the many relevant works related to pedestrian motion prediction. For a more comprehensive overview, Rudenko et al. provide a survey describing various approaches to the human motion trajectory prediction problem [108].

This approach is unique from other works in that I include explicit modeling of aleatoric and epistemic uncertainty within the deep recursive filter framework. I show this combined uncertainty modeling with learned dynamics from a recursive filter framework is imperative to improve state estimation and prediction interval robustness in the presence of out-of-distribution noise. Further, I show that these techniques apply not only to toy problems but also the challenging problem of pedestrian localization using monocular cameras. Finally, I investigate whether properties of a recursive filter can be used as a measure of competency to further improve computational efficiency of the uncertainty estimation.

4.3 State and Uncertainty Estimation

State estimation is a critical problem impacting a wide variety of robotic applications including mapping, localization, pose estimation, and motion planning. These challenging applications are encumbered by issues such as high dimensional observations, partial observability, and noisy measurements. Traditional methods of state estimation, including Kalman and other recursive filters, decouple perception and state estimation by operating directly on low dimensional state representations after the perception pipeline. This limits the ability to develop state estimation techniques that are robust to high sensor noise.

Deep learning techniques have made a significant contribution in many areas including perception, speech recognition, and robotics. Particularly, they offer the ability to operate directly on high dimensional spaces where each layer represents learned features that can be used to perform functions such as classification and regression. While deep learning techniques have demonstrated significant improvements in predictive accuracy, they have demonstrably fallen short in estimating the predictive uncertainty. As demonstrated by Ovadia and Fertig [89], this is particularly exacerbated when evaluating predictive uncertainty under distributional shift. For many applications, including robotics, this is a fundamental limitation as robustness to out-of-distribution noise is critical to improve the safety and reliability of systems when deployed into the real world.

In this section, I study methods that combine deep neural networks with traditional state estimation techniques that improve robustness to noisy input data. I specifically focus on developing techniques that capture aleatoric and epistemic uncertainty [87] to improve prediction interval accuracy in the presence of noisy, out-of-distribution inputs.

The specific focus is along three core research questions:

1. Can principles of recursive filters combined with neural networks improve state estimation accuracy of dynamical systems in the presence of out of distribution noise?

2. Can this approach improve the robustness and performance of predictive uncertainty estimation as the test distribution deviates from the training distribution?

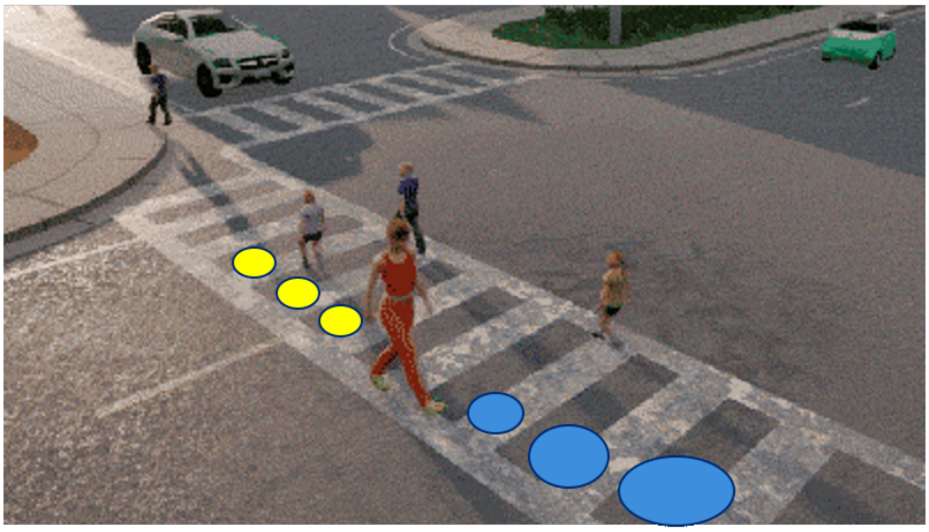


Figure 4.1: The goal of this project is to develop robust state and uncertainty estimation for pedestrian localization by combining elements of deep neural networks, recursive filters and uncertainty estimation.

3. Can properties of the recursive filter provide additional insight regarding expected competency of the trained network?

To study these questions, I evaluate this approach using two experimental environments. The first is a toy problem using a simulated pendulum. The objective is to estimate the pendulum angle in the presence of varying observation noise added to the 24×24 pixel images as input. My goal is to assess the effects of a learned dynamical model and demonstrate robustness during regions of high observation noise.

I then focus on a real world problem by estimating the 3D pedestrian localization from monocular images. For this evaluation, I use the nuScenes [8] dataset consisting of real world driving scenarios with corresponding annotations including pedestrian position. Again, I assess the ability of this model

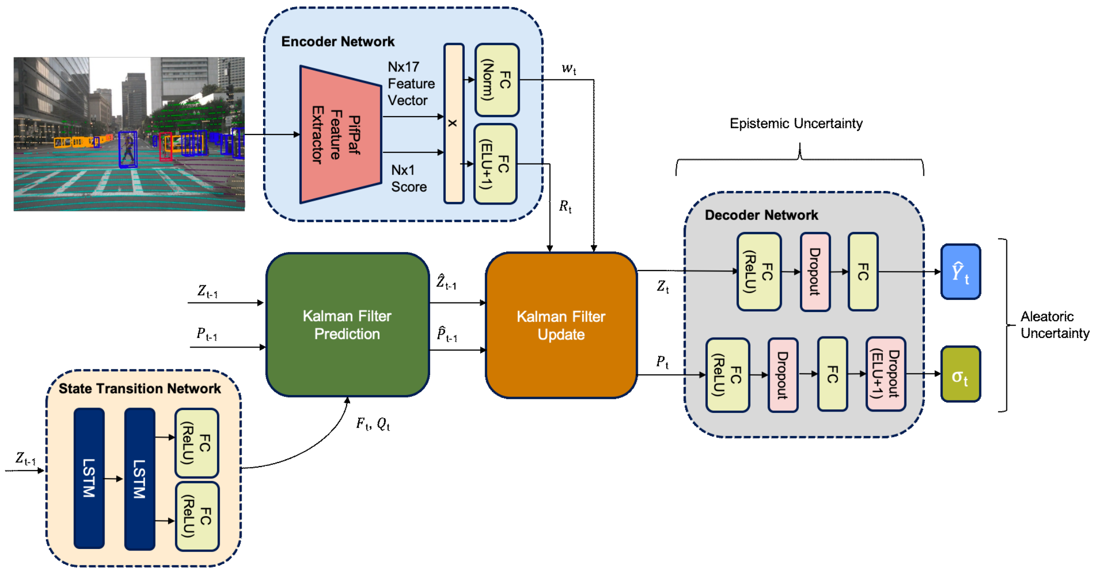


Figure 4.2: This figure describes the overall network architecture for the pedestrian localization experiment. It consists of an encoder network used to learn a latent embedding and measurement covariance matrix, a state transition network used to learn a dynamics model and process covariance matrix and a decoder network used to decode the latent embedding to estimated depth along with capturing aleatoric and epistemic uncertainty.

to provide robust state and uncertainty estimation in the presence of out-of-distribution noise for this more complex, real world scenario.

The following summarizes the main contributions:

1. I develop an architecture, referred to as deep recursive filter (DRF) that combines deep neural networks to learn latent embeddings, state transition models, and associated covariance matrices with a recursive filter operating on the learned latent states.

2. This approach explicitly models aleatoric and epistemic uncertainty as part of the deep recursive filter to improve confidence interval prediction.

3. I show through experimental evidence that my combined approach improves state estimation, confidence interval estimation, and runtime performance on a toy problem and real world pedestrian localization.

4. Further, I show that my approach is more robust to out-of-distribution noise compared to state-of-the-art approaches.

4.3.1 Preliminaries

The objective is to perform state and uncertainty estimation operating directly on high-dimensional observations. Formally, I define the noisy pendulum problem as follows. At given time t, I observe a 24×24 pixel image with noise added to the image. The objective is to estimate the state of the pendulum defined as: $X_t = (cos(\theta_t), sin(\theta_t))^T$ as well as the predicted uncertainty of each state variable, $\sigma_{cos,t}$ and $\sigma_{sin,t}$.

For the pedestrian localization experiments, I define the state of the pedestrian i at time t as $X_{i,t} = (x_{i,t}, y_{i,t}, z_{i,t})$, the Cartesian position of the pedestrian. Similar to [7], I assume a calibrated camera provides the transformation from image coordinates to the x and y Cartesian points. The trained neural network receives a 1600×900 3-channel RGB image and estimates the z_t coordinate corresponding to the depth of the pedestrian using a single monocular image as well as the aleatoric and epistemic uncertainty of the depth estimate, $\sigma_{depth,t}$.

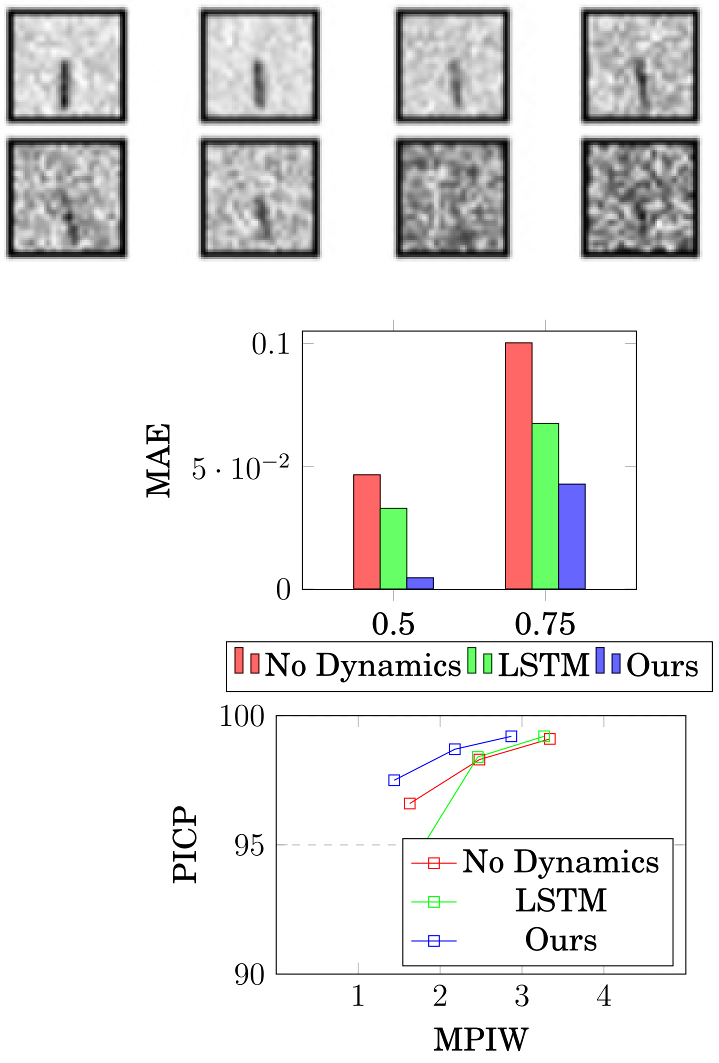


Figure 4.3: (Top) Samples of the noisy pendulum. (Middle) The figure compares the MAE between the DRF approach with the baselines architectures for the noisy pendulum problem for the out-of-distribution noise experiments. (Bottom) This figure compares the PICP metric with the MPIW metric for the three architectures. I show this approach significantly reduces state estimation error and improves to the uncertainty estimation compared to the baselines.

4.3.1.1 Metrics

My objective is to improve state and uncertainty estimation while developing techniques that are computationally efficient. To evaluate the accuracy of state estimation, I measure the Mean Absolute Error (MAE) (Eq. 4.3.1.1) between the predicted and ground truth states. I use the average execution time to compare the computational efficiency of various approaches. To assess the quality of uncertainty estimation, I use the Prediction Interval Coverage Probability (PICP) [109,110] and Mean Prediction Interval Width (MPIW) [109,111] metrics as defined by Eq. 4.3.1.1 and 4.3.1.1. Here, $l(x_i)$ and $u(x_i)$ are the lower and upper bounds of the confidence interval respectively, $\hat{y}_i$ is the point state estimate, y_i is the ground truth, and 1 is the indicator function.

$$\text{MAE} := \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{4.0}$$

$$c_i := 1(l(x_i) \le y_i \le u(x_i))$$

$$\text{PICP}_{l(x),u(x)} := \frac{1}{N}\sum_{i=1}^{N} c_i, \tag{4.0}$$

$$\text{MPIW}_{l(x),u(x)} := \frac{1}{N}\sum_{i=1}^{N} |u(x_i) - l(x_i)| \tag{4.0}$$

Intuitively, the PICP metric measures the percentage of samples where the ground truth falls inside the predicted confidence interval and the MPIW metric measures the average width of the predicted interval. The goal is to maximize PICP while minimizing the MPIW metric.

4.3.2 Approach

This algorithm architecture, inspired by [84] is described in Fig. 4.2 which consists of encoder, decoder and state transition neural networks along with the Kalman Filter (KF) prediction and update steps.

Encoder Network

The encoder network receives a $24 \times 24 \times 1$ grayscale image in the pendulum experiments or a $1600 \times 900 \times 3$ RGB image for the pedestrian localization experiments to produce a latent embedding representing and an associated observation noise covariance matrix. In the pendulum experiment, the encoder network consists of a 5×5 convolutional layer and a 3×3 convolutional layer. This is followed by a fully connected layer to produce a feature vector $\mathbf{x\_in}_{i,t}$. For the pedestrian localization experiments, similar to [7], I first extract PifPaf [112] features that represent a 17×2 dimensional keypoint vector ($x_{i,t}$, $y_{i,t}$) for each pedestrian i along with a confidence score. They keypoint vector and confidence scores are concatenated to produce feature vector $\mathbf{x\_in}_{i,t}$. In both experiments, I pass the feature vector $\mathbf{x\_in}_{i,t}$ to two fully connected layers to generate a latent embedding $\mathrm{w}_{i,t}$ for each pedestrian along with a covariance matrix $\mathbf{R}_{i,t}$ that represents the observation noise. The output features of $Linear_1$ and $Linear_2$ are both 30 dimensions.

$$\mathbf{w}_i^t = Layer\_Norm(Linear_1(\mathbf{x\_in}_{i,t}))$$

$$\mathbf{R}_i^t = Diag[Elu(Linear_2(\mathbf{x\_in}_{i,t})) + 1]$$

State Transition Network

The main goal of the state transition network is to learn a dynamical model in the latent embedding, $\mathrm{w}_{i,t}$, through a learned state transition matrix, $\mathbf{F}_t$ and associated process noise covariance matrix $\mathbf{Q}_t$. I accomplish this by using two LSTM layers followed by a fully connected layer for each output, $\mathbf{F}_t$ and $\mathbf{Q}_t$.

Kalman Filter Prediction and Update

The Kalman filter prediction step follows the principles of traditional Kalman filters where the objective is to produce an a priori estimate of the state and estimate covariance matrix, $\hat{\mathrm{z}}_{t-1}$ and $\hat{\mathbf{P}}_{t-1}$ respectively using the learned state transition matrix. The state transition matrix and associated process noise covariance matrices, F_t and $\mathbf{Q}_t$ are learned by the state transition network. The following operations are performed on the previous latent embedding, z_{t-1}.

$$\hat{\mathbf{z}}_{t-1} = \hat{\mathbf{F}}_t \mathbf{z}_{t-1}$$

$$\hat{\mathbf{P}}_{t-1} = \hat{\mathbf{F}}_t \mathbf{P}_{t-1} \hat{\mathbf{F}}_t^T + \mathbf{Q}_t$$

The KF update step produces a posteriori estimate of the latent embedding and estimate covariance matrix, $\mathbf{z}_{t-1}$ and $\mathbf{P}_{t-1}$ according to the following equations.

$$\tilde{\mathbf{y}}_t = \mathbf{w}_t - \mathbf{H}\mathbf{z}_{t-1}$$

$$\mathbf{S}_t = \mathbf{H}_t \hat{\mathbf{P}}_{t-1} \mathbf{H}_t^T + \mathbf{R}_t$$

$$\mathbf{K}_t = \hat{\mathbf{P}}_{t-1} \mathbf{H}_t^T \mathbf{S}_t^{-1}$$

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \mathbf{K}_t \tilde{\mathbf{y}}_t$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \hat{\mathbf{P}}_{t-1}$$

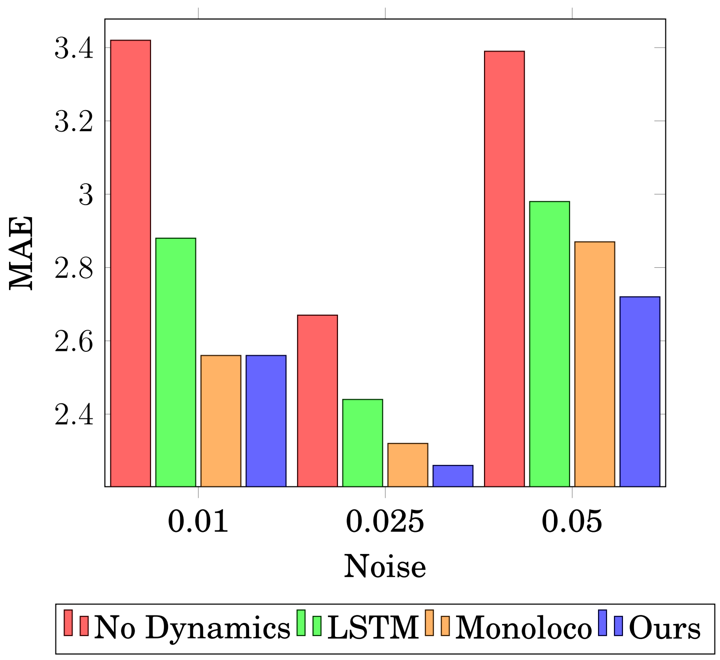


Figure 4.4: Comparison of MAE for pedestrian localization with a changing distribution of noise. In these experiments, the DRF approach performs comparably to Monoloco [7] for in-distribution evaluation and outperforms all baselines for out-of-distribution evaluation.

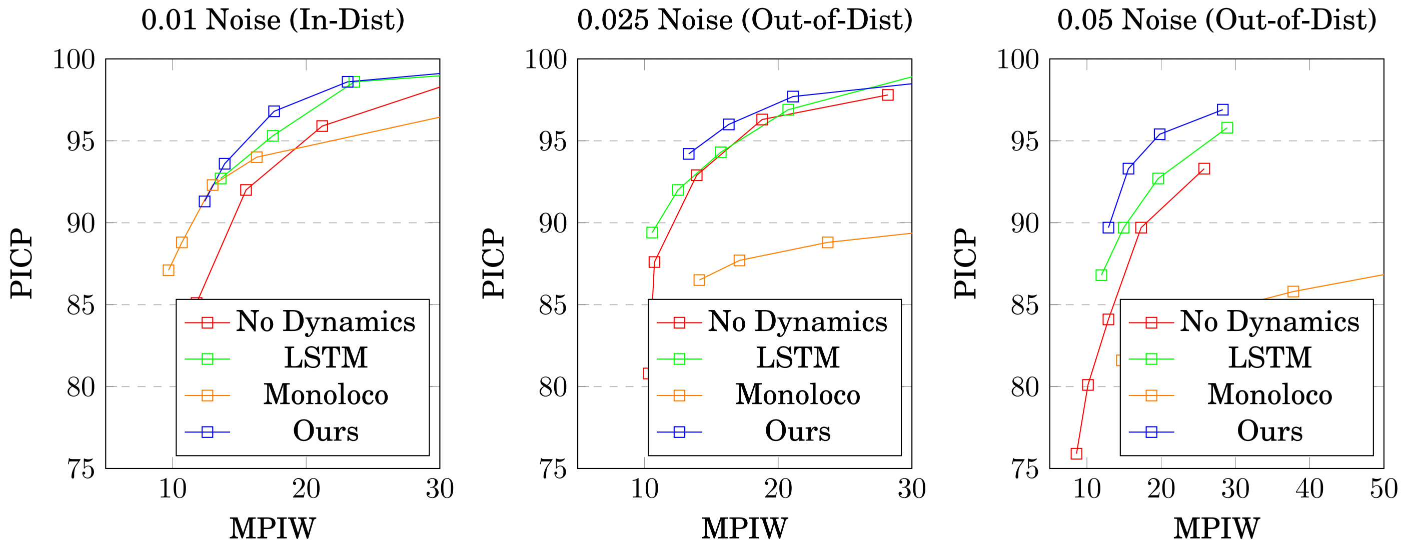


Figure 4.5: A comparison of the PCIP versus MPIW metrics for each algorithm for in and out-of-distribution noise. The data points are generated by varying the dropout rate when computing the epistemic uncertainty. All three algorithms generally show comparable performance for in-distribution samples, however DRF performs better as the noise distribution increases.

Decoder Network

The decoder network has two purposes: 1) receive the posteriori estimate of the latent embedding and covariance matrix to decode an estimated state, 2) to calculate the associated aleatoric and epistemic uncertainties of the state estimate. Aleatoric uncertainty is computed using the mean variance estimation (MVE) [113] technique where the decoder produces two outputs that represent the mean and variance of a Normal distribution that are sampled during inference. To calculate the epistemic uncertainty, I use a stochastic dropout technique [88] where several dropout layers were added to the decoder. During inference, multiple passes of the neural network generate a sample population from which the mean and variance can be calculated. The decoder network can be described as follows:

$$\mathbf{h}_{i,t} = ReLU(Linear_3([\mathbf{z}_{i,t}, \mathbf{P}_{i,t}])$$

$$\mathbf{y}_{i,t} = Linear_4(\mathbf{h}_{i,t})$$

$$\sigma_{i,t} = Elu(Linear_5(\mathbf{h}_{i,t})) + 1$$

Training Details

The entire network is trained using the Pytorch [114] framework with the Adam optimizer [115] with the learning rate set to 10^{-4} for 200 epochs. I use the negative log likelihood (NLL) of a Normal distribution as the loss function computed at the end of a sequence of length S according to Eq. 4.3.2.

$$\mathcal{L} = \frac{1}{S}\sum_{1}^{S}(log(\sqrt{2\pi\sigma_s^2} + \frac{1}{2\sigma_s^2}\sum(y_s - \hat{y}_s)^2) \tag{4.0}$$

4.3.3 Noisy Pendulum Experimental Evaluation

I first evaluate out-of-distribution robustness on the noisy pendulum problem similar to [81,82]. For the training set, I generate 2000 sequences of length 75 corresponding to pendulum motion. The dataset includes a $24 \times 24 \times 1$ image as the input and the corresponding state of the pendulum as defined by $X_t = (cos(\theta_t), sin(\theta_t))^T$. For each sequence, random Gaussian noise was added to the images. I follow the noise generating procedure described in [82] where I vary the maximum amount of noise between 0 and 50% during training.

To measure robustness to out-of-distribution noise, during evaluation, I set the maximum noise threshold to 75% to simulate out-of-distribution noise. I compare against two baselines. The first baseline, referred to as no_dynamics, removes any aspects of a learned dynamical model. Here I map the outputs of the encoder network, w_t and R_t directly to the decoder network and train end-to-end in a similar fashion. The second baseline replaces the Kalman filter elements (e.g., State Transition Network, Kalman Filter Prediction, and

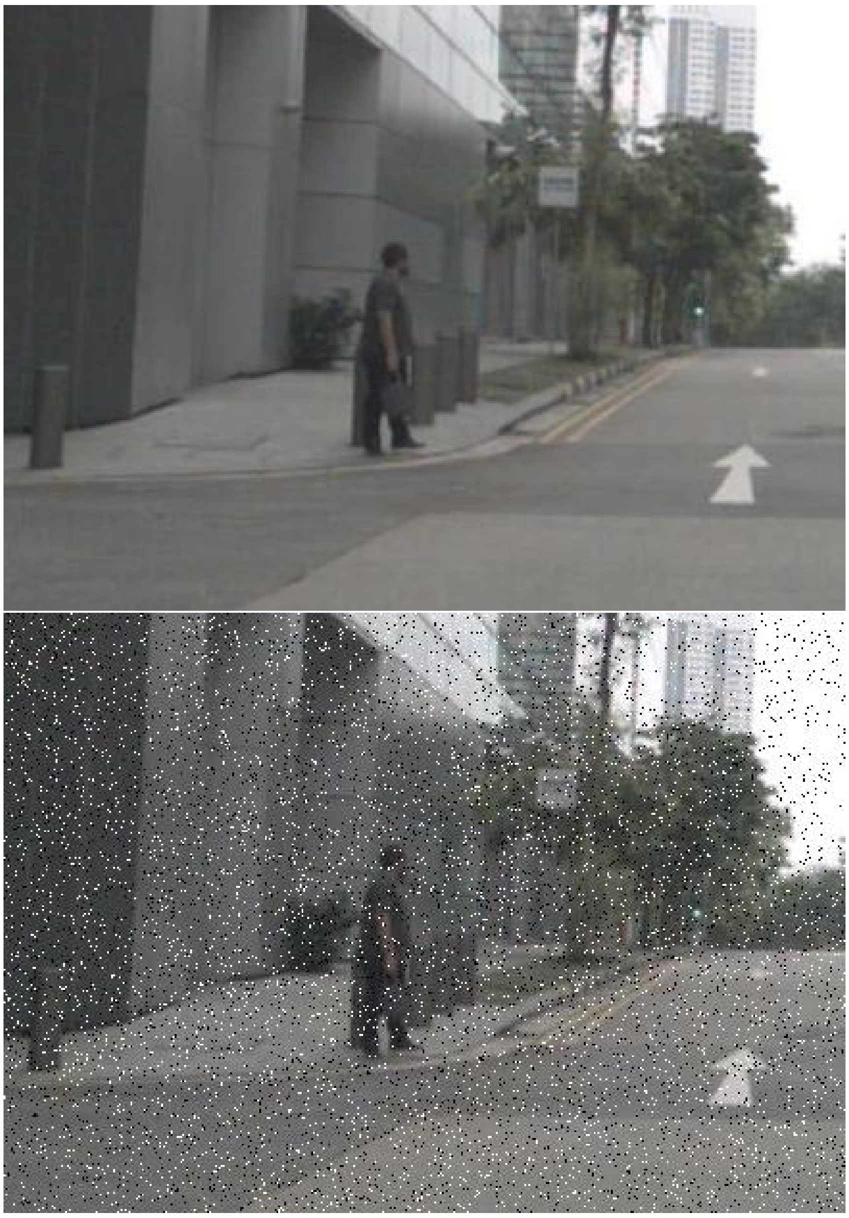


Figure 4.6: (Left) Representative image from nuScenes dataset [8]. (Right) Distorted image with noise added.

Kalman Filter Update steps) with a vanilla LSTM. As demonstrated in Fig. 4.3, I show significant improvement to state estimation in both the in and out-of-distribution noise profile. Further, the recursive filter demonstrates better robustness to predicting confidence intervals.

4.3.4 Pedestrian Localization Experimental Evaluation

I pivot the evaluation to focus on pedestrian localization. For this experiment, I use the nuScenes (part 1) training and validation dataset [8] with tracked pedestrians per key frame. This consists of approximately 3300 images with approximately 3700 pedestrian instances. I add Gaussian noise to these images to represent out-of-distribution evaluation. I trained with a probability of 0.01 that a pixel will be replaced with either a black or white pixel. I then evaluated performance with probabilities of 0.025 and 0.05 to represent out-of-distribution noise (example found in Fig. 4.6)[1].

4.3.4.1 Baseline Comparisons

I compare this algorithm to several baselines for comparison purposes. The first two baselines, no_dynamics and LSTM are similar to the baselines used

[1]I experimented with higher probabilities of noise, however this caused PifPaf detector to miss a significant amount of detections. Future work will bypass this feature extractor layer.

Algorithm	Avg. Time ↓	Speed Up
Monoloco	36.1	1x
Ours	**12.3**	**2.93x**

Table 4.1: The average execution time for 25 forward passes. These numbers are computed after the PifPaf feature extraction which is common to both approaches and measured using an NVIDIA® GTX 1080 GPU.

in the noisy pendulum experiments. The third baseline that I use for comparison is Monoloco [7] which similarly estimates pedestrian localization with uncertainty.

4.3.4.2 State Estimation

I first measure the MAE of the estimated depth of the pedestrian compared to the ground truth provided by the nuScenes dataset. In Fig. 4.4, I compare this approach to the no_dynamics, LSTM and the Monoloco work. When evaluating in distribution, this approach is comparable to Monoloco, with both approaches outperforming the no_dynamics and LSTM baselines. When evaluating out-of-distribution, I show an 18% improvement in the MAE compared to the no_dynamics baseline, 8% improvement compared to the LSTM baseline, and a 4% improvement compared to Monoloco.

4.3.4.3 Uncertainty Estimation

As described in Sec. 4.3.2, I use the PICP and MPIW metrics to assess the quality of the prediction intervals. These results are summarized in Fig. 4.5, where I analyze the trade-offs between PICP and MPIW metrics for each algorithm across in and out-of-distribution noise. For the in-distribution case, all four algorithms exhibit similar performance in uncertainty estimation. When evaluating out-of-distribution, the performance begins to diverge with this approach showing an distinct improvement to the other methods in particularly high out-of-distribution noise samples.

4.3.4.4 Computational Efficiency

An important criteria is to allow uncertainty estimation to run in real time. For that purpose, I also evaluate the run time performance of this approach as summarized by Table 4.1. I demonstrate an approximately 3x improvement largely due to the simpler decoder network used to perform epistemic uncertainty.

4.4 Pedestrian Prediction

As I continue to develop mobile robots to support various human activities ranging from security surveillance to warehouse automation, it is critical for

Algorithm	PICP ↑	MPIW ↓	Avg. Time ↓
Pend - (no dropout)	79.3	0.04	2.48
Pend - (random)	86.3	0.30	17.1
Pend - (thresholded by Kalman gain)	92.1	0.24	16.8
Pend - (always dropout)	92.9	0.48	28.23
Ped - (no dropout)	86.0	10.95	1.63
Ped - (random)	87.4	11.5	4.25
Ped - (thresholded by Kalman gain)	87.7	11.6	4.06
Ped - (always dropout)	89.7	12.9	11.3

Table 4.2: This table captures the effects of using the Kalman gain as a measure of competency on choosing when to compute the epistemic uncertainty.

these robots to move efficiently and safely around humans. To successfully integrate these robots into human environments, I built on prior research exploring robot navigation in human crowds (e.g., [92]) and drew insights from seamless human navigation through crowded spaces. One way that people accomplish seamless navigation is by anticipating other pedestrians' future movements and adjusting their own behaviors accordingly; for example, people slow down or change directions to avoid collisions [22]. In this work, I seek to computationally realize and evaluate human-inspired movement anticipation and how such anticipation may enhance the quality of robot navigation in terms of success rate and pedestrian comfort.

Specifically, this work makes the following contributions: (1) a novel approach to pedestrian prediction that combines generative adversarial networks with a probabilistic model of intent that achieves performance which matches or exceeds state-of-the-art baseline algorithms on real world datasets, (2) the

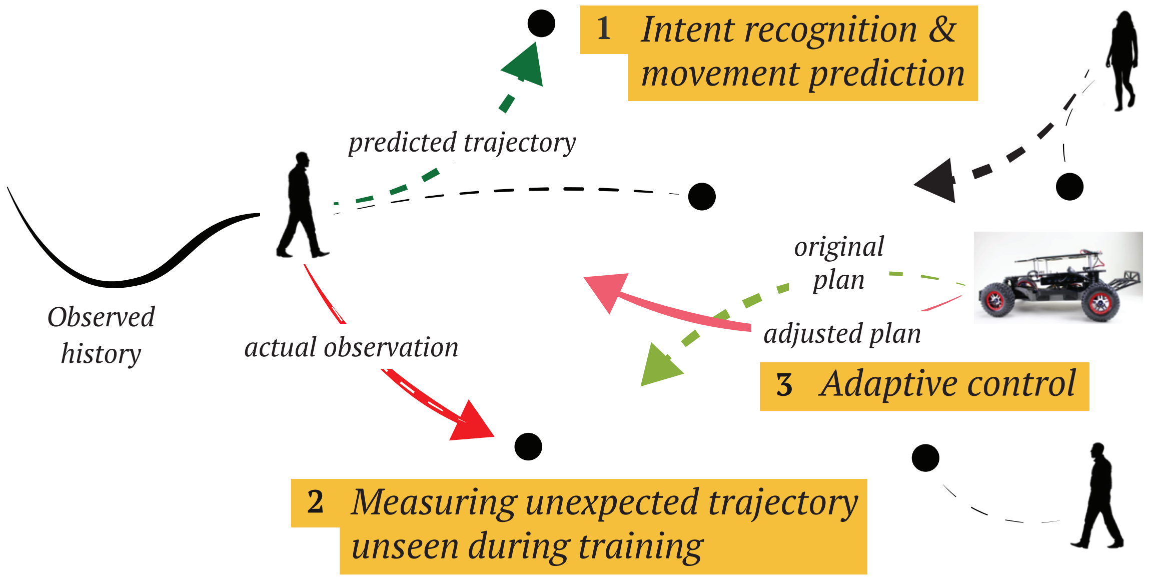

1 Intent recognition & movement prediction
predicted trajectory
Observed history
actual observation
original plan
adjusted plan
3 Adaptive control
2 Measuring unexpected trajectory unseen during training

Figure 4.7: Adaptive crowd navigation policy that uses pedestrian intent and prediction error to adjust the risk profile of a control policy.

ability to use errors in predicted pedestrian motion to detect novel pedestrian behaviors not seen during training, (3) an adaptive policy that adjusts the risk of the robot controller based on detecting novel pedestrian behaviors to minimize collisions.

The remainder of this section is organized as follows. In Sec. 4.4.1, I describe a novel approach to longer term pedestrian prediction that probabilistically reasons about the pedestrian's intent. In Sec. 4.4.2, I compare the prediction results using real world datasets and show comparable or better performance than state-of-the-art baselines.

While each of these papers makes significant contributions to their respective fields, none of the prior work, as far as I am aware, focuses on pedestrian prediction as a measure of uncertainty to inform adaptive crowd navigation

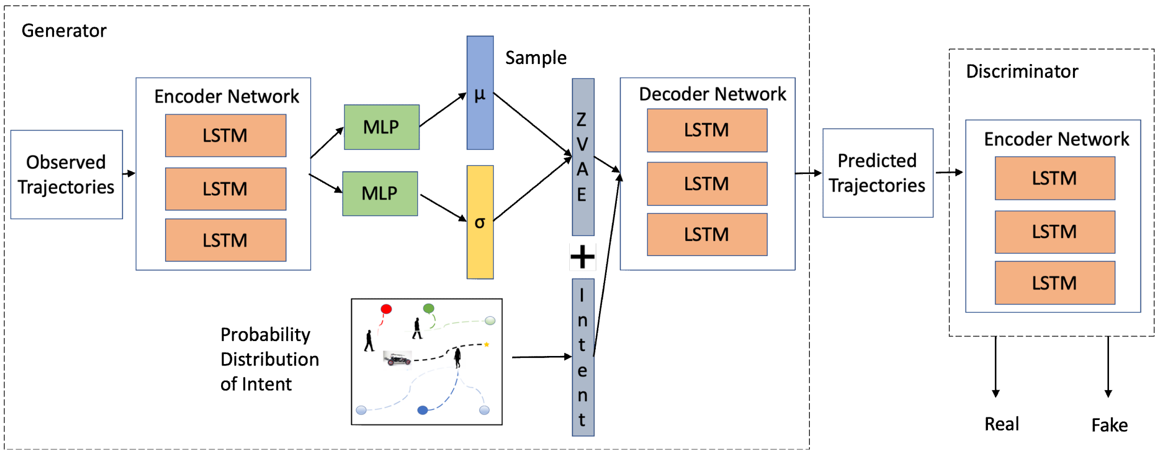


Figure 4.8: The pedestrian prediction network architecture. The generator network consists of a recurrent encoder network, a variational autoencoder, an intent prediction module and recurrent decoder network. The discriminator network consists of a recurrent encoder network to distinguish between real and fake trajectories.

policies.

4.4.1 Problem Formulation

The objective is to estimate future pedestrian trajectory given an observation history of past trajectories. Formally, at given time t, I represent the state of the pedestrian, i as: $X_i^t = (x_i^t, y_i^t)$. I observe pedestrian state during a time window $t = 1$ to $t = T_{obs}$ represented as $X_i^{obs} = [(x_i^1, y_i^1), ..., (x_i^{T_{obs}}, y_i^{T_{obs}})]$. The objective is to predict the pedestrian state from time $t = T_{obs+1}$ to $t = T_{pred}$ represented as $Y_i^{pred} = [(x_i^{T_{obs+1}}, y_i^{T_{obs+1}}), ..., (x_i^{T_{pred}}, y_i^{T_{pred}})]$.

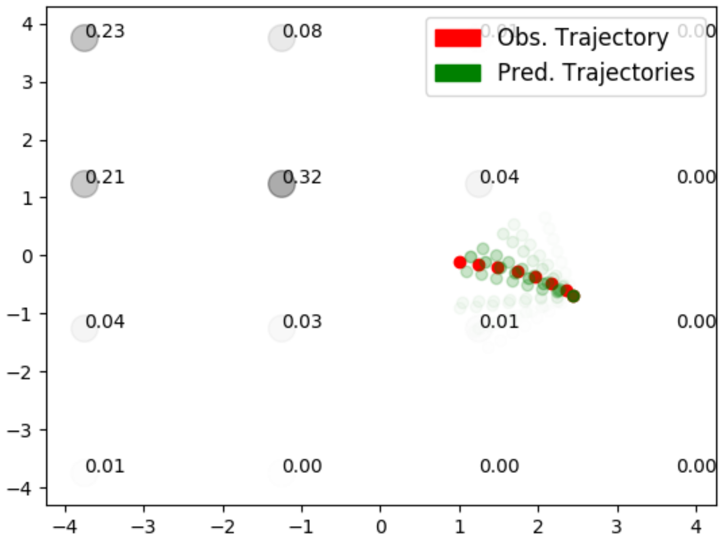


Figure 4.9: Start of the Trajectory.

Neural Network Architecture

The network architecture described in Fig. 4.8 consists of a generator and a discriminator network. The generator network includes a recurrent encoder network, a variational autoencoder, a recurrent decoder network, and an intent predictor. The discriminator network classifies samples as either real trajectories or not socially acceptable [19].

Encoder Network

The recurrent encoder network consists of a linear spatial embedding layer with a ReLU activation layer, $\alpha(\cdot)$, followed by an LSTM layer, where W_e and

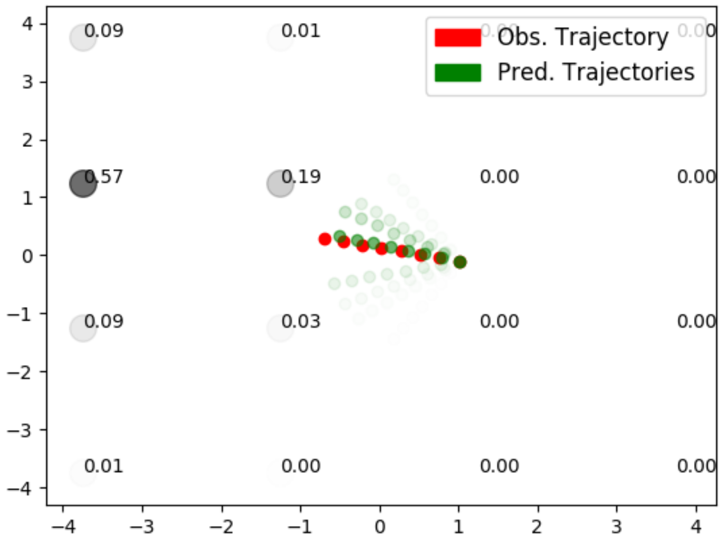


Figure 4.10: Middle of the Trajectory.

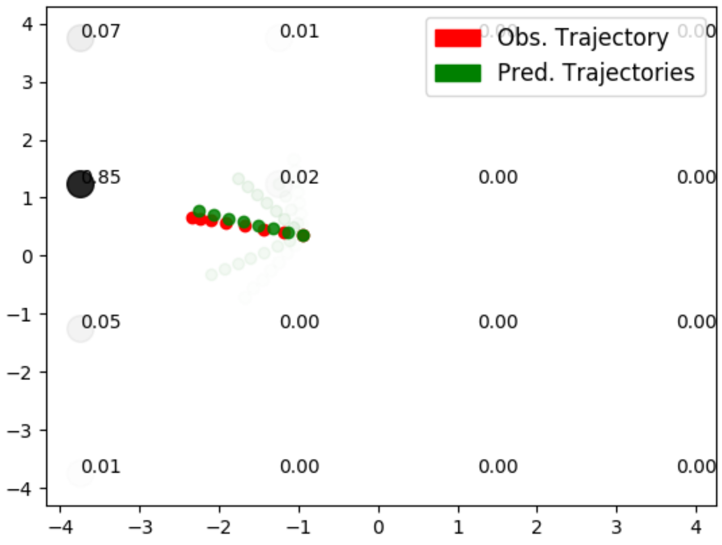


Figure 4.11: End of the Trajectory.

W_{lstme} are weights of the embedding layer and LSTM, respectively.

$$h_i^t = \alpha(X_i^t, W_e)$$

$$p_i^t = LSTM_e(p_i^{t-1}, h_i^t, W_{lstme})$$

I then use two fully connected linear layers (MLPs) to produce latent distribution parameters μ and σ, and sample from this distribution represented as z_{vae} to generate sample diversity, where $W_\mu, W_\sigma, b_\mu,$ and b_σ are the weights and biases of the fully connected layers, respectively.

$$\mu_i^t = W_\mu p_i^t + b_\mu$$

$$\sigma_i^t = \log(\exp(W_\sigma p_i^t + b_\sigma) + 1)$$

$$z\_vae_i^t = \mu_i^t + \sigma_i^t \cdot \epsilon; \epsilon \sim \mathcal{N}(0, 1)$$

Intent Recognition

In this section, I describe the probabilistic model of intent recognition and how navigation intent can be combined with latent embeddings from the recurrent encoder network to improve longer term prediction of pedestrian trajectory.

Inspired by [116], I use a Bayesian approach to estimate the probability of a desired goal, g_i^t, of the pedestrian i based on a past observation history, X_i^{obs} spanning $t = 1$ to $t = T_{obs}$, as described by Equation 4.4.1. $P(g_i^t|X_i^{obs})$ is the posterior probability of each goal, g given an observation history X_i^{obs}. $P(g_i^t)$ is the prior probability of a pedestrian i choosing a given goal at time t. $P(X_i^{obs}|g_t^t)$ is the likelihood probability of observing X_i^{obs} given g_t^t and modelled as a Gibbs measure described by Equation 4.4.1. Here, $E(X_i^{obs}|g_i^t)$ is an energy function that I set equal to distance between the observed trajectory and the shortest trajectory to the goal. β is a hyperparameter that adjusts the landscape of the resulting probability distribution and $Z(\beta)$ is a normalizing constant.

$$P(g_i^t|X_i^{obs}) = \frac{P(g_i^t)P(X_i^{obs}|g_i^t)}{P(X_i^{obs})} \propto P(g_i^t)P(X_i^{obs}|g_i^t) \tag{4.0}$$

$$P(X_i^{obs}|g_i^t) = \frac{1}{Z(\beta)}\text{exp}(-\beta E(X_i^{obs}|g_i^t)) \tag{4.0}$$

In this work, I assume $P(g_i^t)$ is a uniform distribution and I set β to 0.5. I explored two ways to represent a discrete set of goals: (1) inferring the goals directly from the data and (2) generating a uniform set of goals in a grid-like pattern. I decided on using a grid-like representation of goals as it affords greater generalization to diverse scenarios. In this work, I generated a uniform 4×4 grid of targets to illustrate the approach. Figs. 4.9, 4.10, and 4.11 show the distribution landscape across the goals based on the observed trajectory in red.

Green trajectories represent ideal paths to each of the targets with the opacity equal to the resulting probability $P(g_i^t|X_i^{obs})$. In these figures, there initially is significant uncertainty regarding the end goal of the pedestrian; however, as the pedestrian navigates closer to the target, the uncertainty decreases towards a specific goal.

The resulting 16 dimensional probability distribution is concatenated to the z_i^t feature vector sampled from the VAE prior to the decoder network of the generator.

$$z_i^t = [z\_vae_i^t, P(g_i^t|X_i^{obs})]$$

Decoder Network

The goal of the decoder network is to use the latent embedding from the encoder network combined with the intent distribution to generate prediction samples. The decoder network used is similar to that in [19] and consists of linear and recurrent layers used to generate pedestrian predictions. The hidden state of the LSTM decoder, p_i^t is initialized to z_i^t. Here W_d, W_{lstmd} and W_o are weights, and ϕ is a fully connected layer.

$$dh_i^t = \alpha(X_i^{t-1}, W_d)$$

$$p_i^t = LSTM_d(p_i^{t-1}, dh_i^t, W_{lstmd})$$

$$\hat{X}_i^t = \phi(p_i^t, W_o)$$

Loss Functions

During training, I use a combination of the mean squared error (MSE) between the ground truth trajectory and the predicted trajectory, adversarial loss, as well as KL-divergence loss from a unit Gaussian distribution for the variational autoencoder as the loss function. Similar to [19], I also experiment with variety loss during training where for each scene I generate k possible outputs and choose the lowest MSE to generate diversity in the samples.

4.4.2 Pedestrian Prediction Experiments

I performed a series of experiments with real world datasets and compared the performance with several state-of-the-art baseline algorithms. In particular, I used two widely used, publicly available repositories—ETH [23] and UCY [1]—consisting of 5 unique datasets (ETH, Hotel, Univ, Zara1, and Zara2) with 4 scenes. The datasets include pedestrian motion with a top down view and annotated pedestrian positions with respect to the world frame.

Training Details

The dimension of the hidden state for the encoder, p is 16. The dimension of the decoder network's hidden state, z is 32 including the 16 dimensions representing probability of navigation intent. I trained for 200 epochs with a batch size of 64 using the Adam optimizer with the initial generator network learning rate set to 0.0001 and the initial discriminator network learning rate set to 0.001.

Baseline Implementations

I compare the algorithm to several baselines representing unique solutions to the pedestrian prediction problem. These baselines include a linear regression algorithm (Linear) that minimizes least square error to estimate parameters of a linear model, a simple LSTM model, (S-LSTM) which is an LSTM model with a social pooling layer [117], Social GAN (SGAN) which uses a GAN in combination with an S-LSTM [19], and finally SoPhie which uses scene contextual information to make predictions [20].

*Metrics

Similar to [19, 20, 118], I use the average displacement error (ADE) and final displacement error (FDE) metrics to compare this approach to existing baselines. The ADE (in meters) is the L2 distance between the ground truth and predicted pedestrian trajectories for each trajectory point. The FDE (in meters) is the final displacement distance between the last point in the predicted trajectory and the ground truth. I use a leave-one-out approach where I train on 4 of the datasets and test on one.

Prediction Results

Table 4.3 summarizes the ADE and the FDE for the 5 datasets described above when observing 8 timesteps (3.2 sec) and predicting 12 timesteps (4.8 sec) into the future. I use similar notation from [19], kV(+IR)-N, where k represents the samples for variety loss, N represents the number of samples taken during test time and +IR represents whether intent recognition is used as part of the prediction. Sample diversity in this sense [19] allows the learning algorithm to produce k predictions and choose the prediction with the smallest MSE to encourage diversity. I compare the results with and without sample diversity (where $k = N = 1$). This approach results in better than or equal ADE and FDE when compared to the baselines for 8 out of 10 experiments without sample diversity and 7 out of the 10 experiments with sample diversity. In addition, this approach achieves best performance when averaging across the 5 datasets for both the ADE and FDE. These results demonstrate the advantages of estimating a probabilistic interpretation of intent and explicitly using this estimate when making longer term predictions of pedestrian trajectories.

Met	**DS**	**Lin**	**LSTM**	**S-LSTM**	**Soph**	**SGAN**				**Ours**			
					1V-20	1V-1	1V-20	20V-20	20VP-20	1V-1	1V+IR-1	1V+IR-20	20V+IR-20
ADE	ETH	1.33	1.09	1.09	0.70	1.13	1.03	0.81	0.87	0.96	0.85	0.77	**0.69**
	HOTEL	**0.39**	0.86	0.79	0.76	1.01	0.90	0.72	0.67	0.60	0.48	0.42	**0.39**
	UNIV	0.82	0.61	0.67	0.54	0.60	0.58	0.60	0.76	0.55	0.53	**0.51**	0.56
	ZARA1	0.62	0.41	0.47	**0.30**	0.42	0.38	0.34	0.35	0.45	0.41	0.36	0.35
	ZARA2	0.77	0.52	0.56	0.38	0.52	0.47	0.42	0.42	0.38	0.33	**0.30**	0.31
	AVG	0.79	0.70	0.72	0.54	0.74	0.67	0.58	0.61	0.59	0.52	0.47	**0.46**
FDE	ETH	2.94	2.41	2.35	1.43	2.21	2.02	1.52	1.62	1.85	1.80	1.66	**1.42**
	HOTEL	**0.72**	1.91	1.76	1.67	2.18	1.97	1.61	1.37	1.18	1.04	0.94	0.79
	UNIV	1.59	1.31	1.40	1.24	1.28	1.22	1.26	1.52	1.17	1.13	**1.09**	1.17
	ZARA1	1.21	0.88	1.00	**0.63**	0.91	0.84	0.69	0.68	0.94	0.87	0.79	0.74
	ZARA2	1.48	1.11	1.17	0.78	1.11	1.01	0.84	0.84	0.79	0.72	**0.65**	0.66
	AVG	1.59	1.52	1.54	1.15	1.54	1.41	1.18	1.21	1.19	1.11	1.03	**0.96**

Table 4.3: Average Displacement Error (ADE) and Final Displacement Error (FDE) in meters for $t_{pred} = 12$ timesteps. This method matches or outperforms state-of-the-art and baseline methods by explicitly estimating intent as a probability distribution of possible goals (lower is better).

4.5 Adaptive Crowd Navigation

I now extend the pedestrian prediction algorithm to enable adaptive crowd navigation policies for mobile robots. I conjecture that errors in pedestrian prediction can serve as a measure of policy uncertainty. Specifically, as I encounter distribution shift of pedestrian motion, the error in pedestrian prediction can serve as an effective method to detect novel pedestrian motions not seen during training. I believe that detecting novel pedestrian motion profiles is a cue to switch to a risk averse control policy and by doing so will result in fewer collisions.

In the development of adaptive navigation, I leverage the CrowdSim simulation environment provided by [99]. This environment provides the ability to model pedestrian motion using an optimal reciprocal collision avoidance (ORCA) model [94]. Pedestrian behavior can be modelled using parameters such as preferred velocity, the maximum distance and time to take into account neighboring agent behavior, pedestrian's radius, and maximum velocity. In addition, CrowdSim provides an OpenAI Gym like environment [119] to experiment with reinforcement learning based policies controlling a robot's actions to reach a target goal while avoiding obstacles.

The state space of the environment follows that of [99, 120] and consists of the following parameters with respect to the robot position as the origin and the x-axis pointing towards the goal: distance from robot position to goal, robot's preferred velocity, actual velocity and radius. For each pedestrian, the state includes position, velocity, radius, and distance between pedestrian and robot.

The action space assumes a nonholonomic unicycle kinematic model for the robot agent and consists of 3 discrete speeds and 6 discrete rotation angles for a total of 18 actions.

In order to learn a policy that allows the robot to successfully reach the target while avoiding collisions with other pedestrians, I use the same reward definition as [98, 99]:

$$R(\mathbf{s}, \mathbf{a}) = \begin{cases} -0.25 & \text{if} \quad d_{min} < 0 \\ -0.1 - d_{min}/2 & \text{else if} \quad d_{min} < 0.2 \\ 1 & \text{else if} \quad \text{robot reached goal} \\ 0 & \text{o.w.} \end{cases}$$

where d_{min} is the minimum distance separating the robot and the humans during the previous timestep.

I train two navigation policies, a risk averse and an aggressive policy. The aggressive policy consists of a preferred velocity of 2.0 m/s, and the risk averse policy is limited to 1.0 m/s [2]. I first assess state-of-the-art crowd navigation

[2]For the risk averse policy, I first considered training a policy where the penalty for collision was significantly increased, however, changing the reward function would make comparisons with existing work difficult.

policies' performance in the presence of a changing distribution of pedestrian motion. I trained both CADRL [120] and SARL [99] with preferred robot velocities set to 2.0 m/s with a static pedestrian motion model. The starting and ending positions of the pedestrians were sampled uniformly inside a square of width 10 meters. The policies were pretrained using imitation learning using ORCA similar to [99] and were subsequently trained using a value iteration network for 10000 steps. I then evaluated the performance of these policies on test data with a shifted distribution of pedestrian motion uniformly sampled from the parameters described in Table 4.4. The objective for this test is to show the limitation in prior works' ability to handle distribution shifts from unseen pedestrian motion during training.

I then conducted a series of experiments with an adaptive control policy using various methods of novelty detection of new pedestrian behaviors.

I first evaluate with a traditional, non-deep learning based approach using a one-class SVM with a radial basis kernel [121]. I train the one-class SVM algorithm based on the fixed pedestrian motion profile and evaluate its ability to detect novel distributions of pedestrian motion data. I then conduct several experiments using deep learning based approaches for novelty detection including Social GAN and the intent-aware pedestrian prediction algorithm. The goal is to demonstrate the benefits that a higher performing pedestrian prediction algorithm can have on reducing collisions in an adaptive crowd navigation policy.

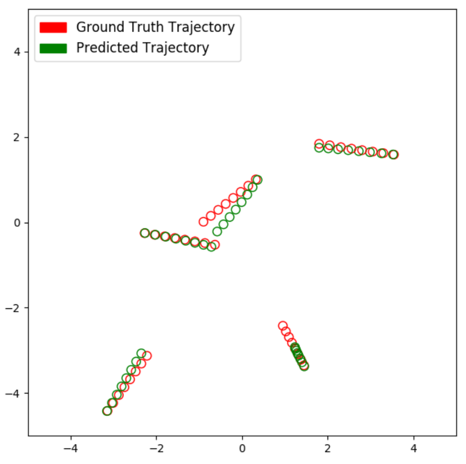
I trained both Social GAN and the prediction algorithm with the same fixed pedestrian motion profile that I trained the original policies. I then tested pedestrian prediction with a changing distribution of pedestrian motion while allowing the robot to navigate. I compute an estimate of novelty by thresholding the prediction error, as measured using the FDE, by a value α. If the error exceeds α, the policy moves from an aggressive behavior to a risk averse policy with the goal of avoiding collisions. The value of α was chosen by computing the mean and standard deviation of the FDE in the training set. In the experiments, α was set to 3 standard deviations from the mean to eliminate outliers.

Table 4.4: ORCA Model Parameters

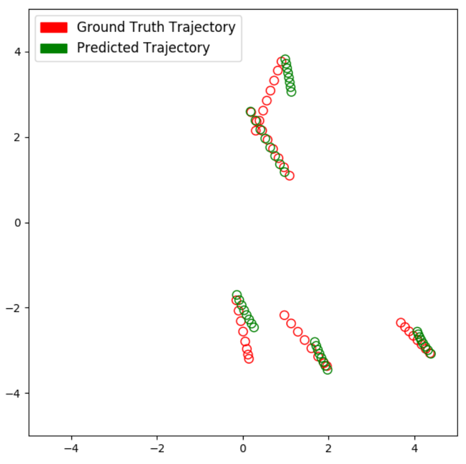
Parameters Name	**Min Value**	**Max Value**
Preferred Velocity	0.5	2.0
Radius	0.2	0.8
Neighbor Distance	2.0	20.0
Time Horizon	0.1	5.0

4.5.1 Quantitative Analysis

The primary metrics I used for comparison are the number of successful trials, number of collisions, the average navigation time, the discomfort level, and the average reward. The discomfort level is defined as the frequency of



(a) Example prediction with low uncertainty

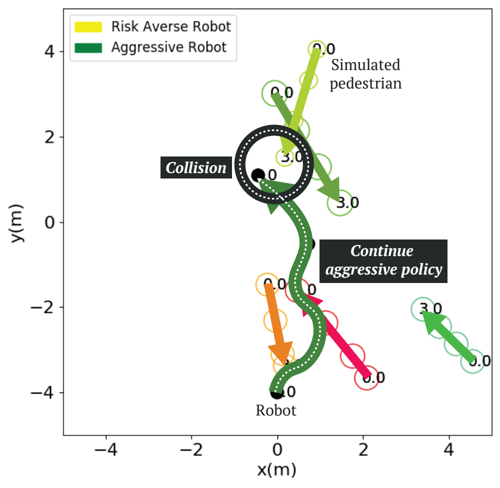


(b) Example prediction with high uncertainty

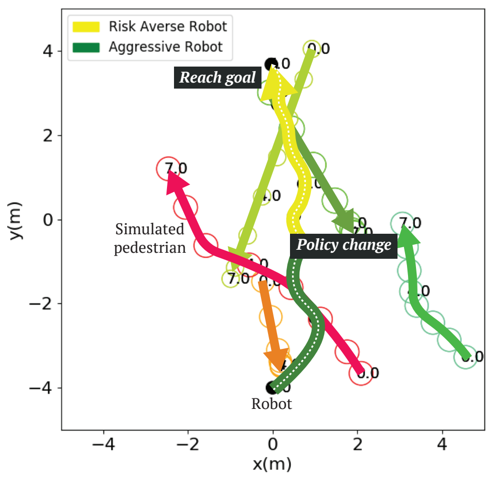
Figure 4.12: This figure provides representative examples of pedestrian predictions with low and high uncertainty.

the separating distance being less than the desired separation distance, in this case 0.2 m. The results after running 500 test cases are reported in Table 4.5. I compare various methods of crowd navigation denoted by $method - p$ where p is the number of pedestrians in the scene and A indicates an adaptive policy. In these experiments, the starting and goal positions, and pedestrian motion profiles are all randomly initialized for each of the 500 test cases; however, these parameters are consistent across the various methods to create a fair comparison.

The first two rows show the performance of CADRL and SARL policies where I train and test without changing the distribution of the pedestrian motion profile. I then evaluate the algorithms' ability to avoid collisions when



(a) Example of non-adaptive policy colliding with pedestrians



(b) Example of adaptive policy successfully reaching target while avoiding pedestrians

Figure 4.13: This figure provides representative examples of trajectories of the robot and pedestrians navigating to their desired goals. The non-adaptive policy continues an aggressive motion resulting in collision. The adaptive policy reverts to a risk averse strategy allowing the robot to reach the goal.

faced with novel pedestrian motion and show that both the CADRL and SARL policies have a significantly higher collision and discomfort rates. The number of collisions for CADRL and SARL increase by 35 and 65, respectively.

The non-deep learning based approach detects novel pedestrian motion using a one-class SVM and reduces the number of collisions by 2 compared to the non-adaptive SARL algorithm.

Using Social GAN as the pedestrian prediction algorithm for novelty detection had a significant improvement compared to the one-class SVM by further reducing the number of collisions by 20.

Our intent-aware pedestrian prediction algorithm for novelty detection resulted in best performance across almost all of the metrics. Using this approach, I was able to further reduce the number of collisions by 5 compared to Social GAN and overall by 30 compared to the non-adaptive SARL policy. I do this while also demonstrating best performance in overall discomfort rate and overall reward. Further, I show that the benefits of this approach scale as the number of pedestrians in the scene increase.

4.5.2 Qualitative Analysis

I further study the qualitative aspects of this approach. In Fig 4.12a, I show an example where the pedestrian prediction algorithm has low FDE hence has high confidence and correctly decides to maintain a high risk, fast navigation policy. Conversely, in Fig. 4.12b, I show an example where the FDE in pedestrian motion is high. This situation is flagged as a novel pedestrian behavior and in this example, a risk averse policy is selected resulting in an avoided collision.

In Fig 4.13a and Fig. 4.13b, I also show example trajectories of the adaptive and non-adaptive policies. In Fig 4.13a, with the non-adaptive policy, even though the pedestrian prediction error is high, the default aggressive policy continues and the robot eventually collides with a pedestrian within 3 seconds. In contrast, using the adaptive policy as shown in Fig. 4.13b, I detect novel pedestrian behavior and instead modify the policy to a risk averse controller. This adaptation causes the robot to reduce its velocity in real time preventing a near collision from occurring. Videos of this behavior and additional examples can be found in the supplemental material of this paper.

4.5.3 Hardware Experiments

I further assess the real world applicability of the algorithms by evaluating in a proof-of-concept physical test environment (Fig. 4.14). The physical test bed consists of the MIT Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot (RACECAR) navigating through several pedestrians. This is the same platform used in the experiments in Chapter 3 and consists of a Hokuyo UST-10LX LiDAR, Sparkfun IMU, the Traxxas 1/10-scale

Table 4.5: Quantitative Analysis of Collision Avoidance

Method	**Dist. Shift**	**Succ.**	**Coll.**	**Time Out**	**Nav. Time**	**Disc. Rate**	**Avg. Rwd.**
CADRL-5	N	455	45	0	4.48	2.02	0.349
SARL-5	N	490	5	5	4.61	0.99	0.389
CADRL-5	Y	420	80	**0**	**4.52**	3.53	0.296
SARL-5	Y	425	70	5	4.62	2.27	0.303
SVM-A-5	Y	426	68	6	5.34	2.14	0.331
SGAN-A-5	Y	445	45	10	6.31	2.03	0.386
Ours-A-5	Y	**450**	**40**	10	6.74	**1.98**	**0.409**
SARL-10	Y	388	99	13	**5.21**	2.62	0.234
Ours-A-10	Y	**444**	**54**	**2**	8.49	**2.18**	**0.330**
SARL-15	Y	290	205	5	**5.30**	4.89	0.115
Ours-A-15	Y	**366**	**132**	**2**	8.69	**4.20**	**0.212**
SARL-20	Y	172	324	4	**5.27**	6.70	-0.017
Ours-A-20	Y	**262**	**237**	**1**	8.65	**6.29**	**0.066**

chassis and an onboard NVidia Jetson processor with GPU.

I use a leg detector algorithm based on an SVM classifier to detect pedestrians with respect to the camera frame and a custom SLAM library to generate maps and estimate robot position. Using the pose of the robot with respect to the global frame and the pedestrian with respect to the robot, I transform the pedestrians to a global coordinate frame allowing us to run the trained navigation policies directly from simulations without requiring further training on the physical hardware. As part of this hardware demonstration, I was able to verify that the trained policy can transfer from simulation to the physical robot, execute in real time, operate on noisy sensors and is able to successfully reach its goal while navigating around pedestrians to avoid collisions.

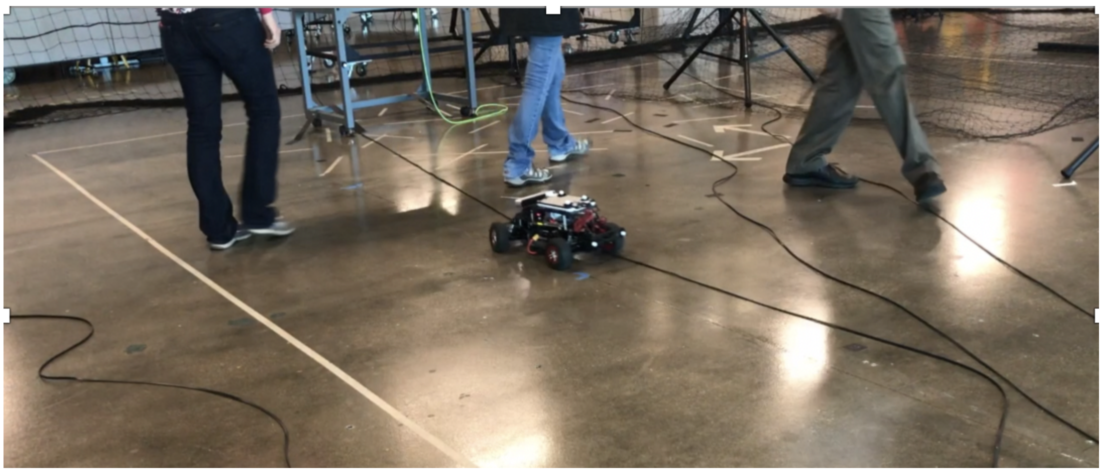


Figure 4.14: Hardware demonstration using MIT Racecar navigating around moving pedestrians.

4.6 Discussion

In this chapter, I focus on developing techniques that allow mobile robots to navigate in the presence of pedestrians. I first focus on pedestrian state and uncertainty estimation including accurate and robust confidence interval prediction in presence of out-of-distribution noise. I show that by modeling the recursive filter framework within a neural network architecture and capturing aleatoric and epistemic uncertainty, I can significantly improve state estimation and the robustness of uncertainty estimation. In the pedestrian localization scenario, I show an average of 4% improvement to the MAE across the varying noise distributions compared to next highest performing baseline and 18% improvement compared to the no_dynamics baseline. Further, in the presence of distributional shift, I show higher percentage of samples found in the prediction interval while also reducing the overall prediction interval width. Finally, I show this technique is more computationally efficient as it computes aleatoric and epistemic uncertainty with approximately 3x performance improvement compared to the baseline.

I then shift the focus to pedestrian prediction based on intent reasoning. I develop an algorithm that combines intent prediction with a history of past kinematic trajectories to estimate future motion. Leveraging widely used pedestrian datasets, I present empirical evidence (Table 4.3) showing the importance of having an explicit, probabilistic representation of the intent in longer term prediction. With this approach, the pedestrian prediction algorithm is able to show improvements to the average displacement and final displacement errors compared against several state-of-the-art algorithms.

Finally, I focus on developing an RL controller to enhance robot navigation in human crowds. In this chapter, I propose techniques that can use the predicted pedestrian motion to enable adaptive policies by detecting out-of-distribution pedestrian motion. When unexpected pedestrian motions are observed, I modify the policy to a low risk controller with the goal of avoiding collisions. Moreover, while showing how several crowd navigation methods fail to avoid collisions in the presence of novel pedestrian motion not seen during training, this approach demonstrated the best results in terms of most number of successful trials, highest overall reward, lowest number of collisions, and lowest overall discomfort rate.

In regards to state estimation, I believe an interesting area of future work remains in better understanding the learned covariance matrices and leveraging the Kalman gain as a measure of competency to improve uncertainty sampling performance. The intuition is that one would only compute the expensive epistemic uncertainty when the estimated competency is low. Applying this technique on the pendulum experiments, I achieve good results. As shown in Table 4.2, the PCIP metric is comparable to always performing stochastic dropout, while improving the MPIW by 50% and the runtime performance by 40%. However, in the pedestrian localization experiment, I do not see the same performance. I believe this can be attributed to the sources of uncertainty. In the pendulum example, much of the observation uncertainty is represented by image noise and is easily captured by the Kalman gain. In the pedestrian example, there are far more sources of uncertainty beyond observation noise including distance from the camera and occlusions that are difficult to capture in the Kalman gain trained in an end-to-end manner. The planned future work is to better characterize these uncertainties and use regularization techniques to further improve competency estimation with the goal of achieving similar performance as the pendulum example. Another opportunity for future work is to better leverage dynamical models. Here, I believe driving the learned state transition network towards a known dynamics model will improve robustness and sample efficiency and is an area I am continuing to further explore.

In regards to pedestrian prediction and navigation, this approach also has limitations. First, the average navigation time did increase compared to the other approaches, although this increase was expected as the reported times excluded test cases where a collision occurred. Moreover, this approach avoided collisions by reducing the speed of the robot which resulted in an expected increase in average navigation time. Second, while this approach reduced the number of collisions, it did not prevent them entirely. There were instances when even though the prediction was accurate, a collision could still occur, suggesting that pedestrian prediction alone does not capture all aspects of uncertainty involved in dynamic crowd navigation. In these scenarios, it may be beneficial to consider other forms of uncertainty estimation such as bootstrapping [122], stochastic dropout [88,123], and multiple hypothesis loss techniques [124, 125]. Third, this work only focused on modeling pedestrians' navigation intent to enhance the quality of robot navigation. Future work should consider other aspects of pedestrian navigation, such as personality [126] and group interaction, to capture nuances in human navigation. Finally, while navigating in the presence of crowds in an important step forward, crowd navigation typically involves pedestrian moving within dynamic groups. Developing algorithms that can detect group membership and enable socially appropriate navigation has the potentially to greatly improve navigation performance as well as increase the acceptance of the robot in human environments. In Chapter 5, I focus on extending the navigation policy to improve many aspects of robot navigation and social performance in the presence of groups.

In spite of these limitations, I believe leveraging Kalman filter techniques with neural networks allows us to develop better state and uncertainty estimation. In addition, explicitly modelling a probabilistic interpretation of intent has shown to improve accuracy of estimating future pedestrian motion. Further, the use of pedestrian prediction has the ability to detect novel pedestrian behaviors not seen while learning a policy. Finally, I show that detecting novel pedestrian behaviors and adapting the policy can significantly reduce the number of collisions compared to alternative approaches.

Chapter 5

Group-Aware Robot Navigation in Crowded Environments

5.1 Introduction

Mobile robots that are capable of navigating crowded human environments in a safe, efficient, and socially appropriate manner hold promise in bringing practical robotic assistance to a range of applications, including security patrol, emergency response, and parcel delivery. As human movements are fast, dynamic, and following delicate social norms, an increasing body of research has focused on the challenging quest to enable human-aware robot navigation [92,127–129]. For example, prior research has treated humans as dynamic obstacles to avoid collisions (e.g., [130]), investigated strategies to avoid getting stuck in human crowds (e.g., [131]), and explored how to model social norms to allow for socially appropriate robot navigation (e.g., [132–134]). However, prior works have mainly treated people as individual, independent entities in robot navigation.

The majority of people, however, walk in groups [135, 136]; an empirical study showed that up to 70% of pedestrians in a commercial environment walked in groups [137]. Consequently, it is imperative that a mobile robot respects human grouping (e.g., not to cut through a social group) during its navigation in a human environment. In particular, in this work, I consider the problem of a robot interacting with dynamic human groups—people walking together in groups—rather than standing groups that are commonly seen in social events (e.g., [138]). While substantial efforts have been made to model and understand dynamic groups (e.g., [137, 139, 140]), how mobile robots should navigate effectively and appropriately around dynamic human groups is under-explored.

In this chapter, I explore robot navigation in crowds of human groups (Fig. 5.1). The approach is to learn navigation policies that allow the robot to safely reach its desired goal while minimizing impact to individual and groups of pedestrians. The contributions include:

- A reinforcement learning (RL) algorithm that combines robot navigation performance and group-aware social norms for learning a robust policy;

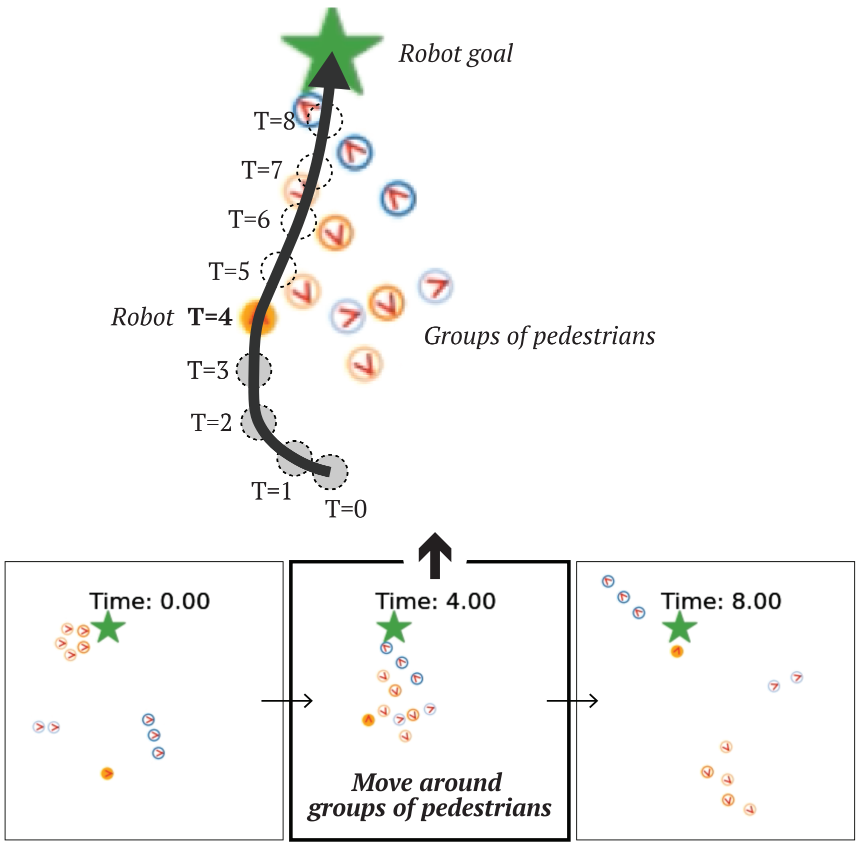


Figure 5.1: The objective of this work is to learn a navigation policy that allows the robot to safely reach its goal while minimizing impact to individual and groups of pedestrians.

- A novel reward function that uses the convex hull of a group (Fig. 5.2) as the group space to minimize impact to pedestrian groups and improve navigation performance;
- Software extensions to the CrowdNav simulation environment [9] to support social navigation research; and
- Experimental results that demonstrate the efficacy of the learned policy with respect to robot navigation performance, human navigation performance, and maintenance of social norms.

5.2 Prior Work

The goal of human-aware robot navigation is to enable robots to move safely, efficiently, and socially appropriately in natural human environments. To achieve safe and efficient navigation, prior research has investigated reactive methods for motion planning (e.g., [141, 142]) and considered modeling pedestrian intention (e.g., [143]). To realize social appropriateness, previous works have explored learning from human data (e.g., [97, 132, 144–147]) and used handcrafted rules as planning constraints (e.g., [133, 148–150]). One notable approach is the Social Force Model (SFM) [151], which is based on the proxemics theory [152] and attempts to model pedestrian social motion using a combination of attractive and repulsive forces. This approach has been adapted and extended for crowd simulation (e.g., [153]) and robot navigation (e.g., [154, 155]).

While prior research on human-aware robot navigation has mostly regarded humans as individual agents, increasing efforts have considered how mobile robots should interact with human social groups. I consider human social groups as two categories—*static* and *dynamic* social groups. Static social groups are commonly seen in social events (e.g., a cocktail party), where people gather together in small groups for conversation. In contrast, dynamic social groups are groups of people walking together, and possibly engaging in conversations during walking, toward shared destinations. Previous research has investigated how to enable robots to recognize (e.g., [156]) and approach (e.g., [157, 158]) static, standing social groups, while taking account of the size and formation of the groups.

Though the detection and modeling of dynamic social groups present additional challenges when compared to static groups, they are critical in enabling socially appropriate robot navigation in human crowds. Prior research has generally explored methods to capture intra-group coherence (e.g., [159, 160]) and inter-group differences (e.g., [161]) in dynamic groups. For instance, salient turn signals in groups of humans that share the same navigation goals can be used to enhance trajectory prediction and subsequently improve the social-awareness in robot path planning [162]. However, it has been shown that group properties may be different in static and dynamic settings. For example, the o-space formation commonly observed in static groups is not necessarily apparent in dynamic groups [163]. To address such difference, a set of dynamic constraints for o-space based on the walking direction and group cohesion was proposed for effective robot navigation [163]. Additionally, dynamic groups also bear unique properties that mobile robots may take advantage of during their navigation in crowded environments. As an example, robots may "group surf" human groups by following their movement flows [164].

Methodologically, modern machine learning techniques have fueled the advancement of human-aware robot navigation. In particular, Recurrent Neural Networks (RNNs) and Generative Adversarial Networks (GANs) have been shown to be able to accurately predict human motion for individuals (e.g., [165, 166]) and groups (e.g., [140]). However, RNN and GAN based methods only predict human motion and do not generate navigation policies for mobile robots. Reinforcement learning approaches are increasingly used for learning navigation policies. For example, prior works have leveraged inverse reinforcement learning to imitate humans and realize socially appropriate movements (e.g., [97, 138]). Deep Reinforcement Learning (DRL) has also been used for robot navigation (e.g., [167, 168]); in particular, attention-based DRL has been demonstrated to capture human-human and human-robot interactions in crowded environments [9, 169].

Different from these prior works, I explicitly include group modeling, rather than a simple consideration of pairwise interactions between individuals in a crowd [168].

In addition, the approach uses a more compact representation of group space by computing a polygon based on the convex hull of the pedestrians instead of the F-formation as in prior works [163, 170]. Furthermore, the approach considers additional metrics when evaluating the navigation performance including group intersections as well as pedestrian and robot performance metrics. Finally, this work differs from the others by using violations of the group space as a reward term to learn socially appropriate movements around groups of pedestrians.

5.3 Preliminaries

5.3.1 Problem Formulation

The main objective is to learn a controller that allows a robot to navigate to a desired goal while maintaining social norms and avoiding collisions with groups of pedestrians. I formulate this approach using reinforcement learning (RL) to learn a policy to meet the objectives stated above. In this form of a Markov decision making process, the robot uses observations to generate a state vector, **S**, and chooses an action, **A**, that maximizes expectation of the future reward, **R**.

5.3.2 State and Action Space

The state space, **S**, consists of observable state information for each pedestrian i, represented as $\mathbf{Ped}_i$ as well as internal state of the robot represented as **Rob** as described by Eq. 5.3.2. Here, p_x and p_y are the x and y coordinates of the position, v_x and v_y are the x and y coordinates of the velocity, rad is the radius of the pedestrian or the robot, g_x and g_y represent the x and y goal positions, v_pref is the preferred velocity and $theta$ is the turn angle.

$$\begin{aligned}\mathbf{Ped}_i &= [p_x, p_y, v_x, v_y, rad],\\ \mathbf{Rob} &= [p_x, p_y, v_x, v_y, rad, g_x, g_y, v\_pref, theta],\\ \mathbf{S} &= [\mathbf{Ped}, \mathbf{Rob}]\end{aligned} \tag{5.0}$$

In the simulation, I assume a robot with holonomic kinematics that receives v_x and v_y commands. The action space is discretized into 5 speeds ranging from 0.2 to 1.0 m/s and 16 rotations ranging from 0 to 2π plus a stop command resulting in 81 possible actions.

5.3.3 CrowdNav Simulation Environment

I leverage the CrowdNav simulation environment [9] for training and evaluating the group aware RL policy. This environment provides a learning and simulation framework that allows us to model scenes of pedestrians and robots interacting while reaching their target goals. I extend this framework by allowing groups of pedestrians to be instantiated with similar starting and end goals. I further leverage the group aware social force model described in the next section to model the motion of the groups of pedestrians as they interact with other pedestrians and the robot.

5.3.4 Group Aware Social Force Model

I use an extended Social Force Model (SFM) proposed by Moussaid et al. [137] to simulate dynamic social groups. In the extended SFM, each individual's motion, as defined in Eq. 5.3.4, is driven by a combination of an attractive force ${f_i}^{des}$ that drives them to a desired goal, the obstacle repulsive forces ${f_i}^{obs}$, the sum of social repulsive forces from other agents $\sum_j f_{ij}^{social}$, and a new group term f_i^{group} defined by Eq. 5.3.4.

$$\frac{dv_i}{dt} = f_i^{des} + f_i^{obs} + \sum_j f_{ij}^{social} + f_i^{group}$$

The group term is defined as the summation of the attractive forces between group members f_i^{att}, the repulsive force between group members f_i^{rep}, and a gaze force f_i^{gaze} that steers the agent to keep the center of mass of the social group within their vision field to simulate with-in group social interactions:

$$f_i^{group} = f_i^{att} + f_i^{rep} + f_i^{gaze} \tag{5.0}$$

A custom Python implementation of the extended Social Force Model was developed [1] for the CrowdNav environment, following the implementation of PEDSIM C++ library [171] and the ROS implementation by Vasquez et al. [172].

5.4 Approach

To evaluate the group aware policy, I extend the existing CrowdNav simulation environment [9] to represent pedestrian motion in groups. I accomplish this by stochastically sampling the number of groups per episode using a Poisson distribution ($\lambda = 1.2$) [173] and then randomly assigning pedestrians to the groups. Each pedestrian within a group has similar start and goal positions. The average number of groups and group size for five pedestrians are 2.5 and 1.96, respectively. For ten pedestrians, the number of groups and group size increase to 4.9 and 2.0, respectively.

[1] https://github.com/yuxiang-gao/PySocialForce

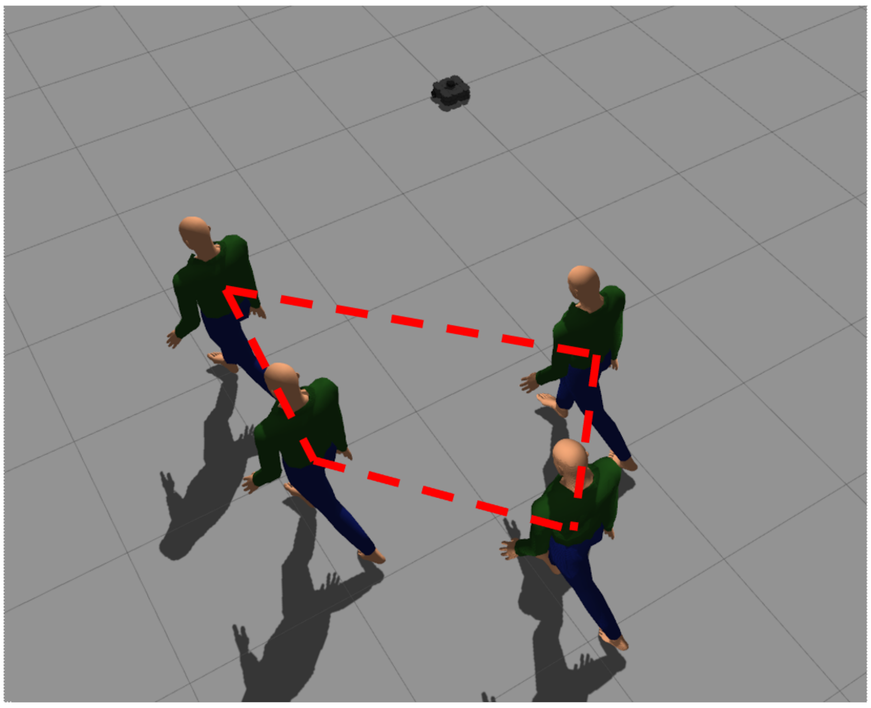


Figure 5.2: During training, I compute the convex hull of pedestrians that are members of a group. I encourage the robot to maintain separation between the robot and the group, which, in turn, leads to a more socially appropriate group navigation policy.

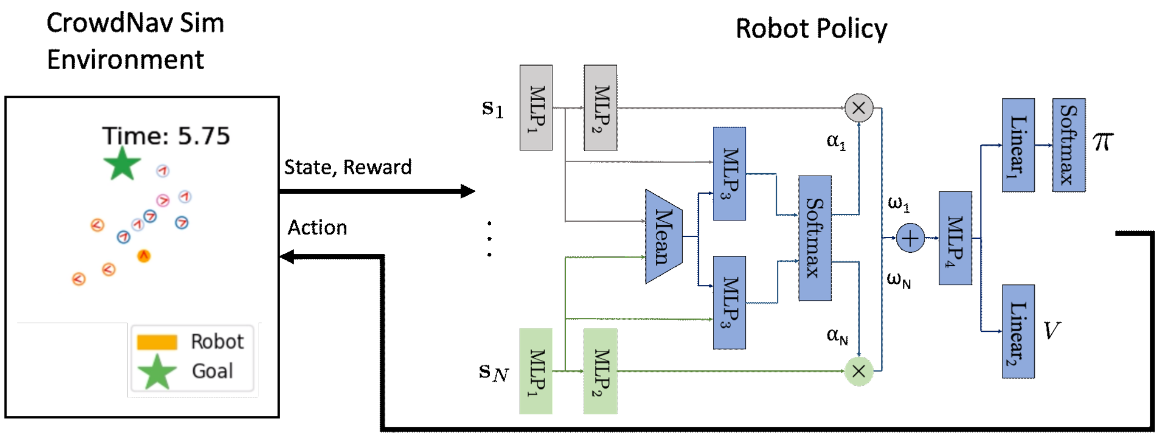


Figure 5.3: On the left is the CrowdNav Simulation Environment [9] that provides pedestrian and robot state information as well as the reward to the policy. The right figure represents the neural network architecture used for the attention-based, actor-critic policy. The pedestrian and robot state vectors are concatenated to represent a pairwise combined state vector; the output of the network are the policy π over potential actions and value V of the current state. The gray and green blocks indicate features from individual pedestrians. The blue blocks indicate aggregate features across pedestrians. The argmax of the policy is chosen as the action, which subsequently is sent to the CrowdNav Simulation Environment to control the robot.

5.4.1 Policy based on Convex Hull of Group

To train the policy, I use a multi-term reward function that encourages the robot to reach its goal while maintaining social norms and avoiding collisions with groups of pedestrians. In particular, I focus on social norms that minimize discomfort to individuals and discourage intersections with a group of pedestrians.

The reward function is given by Eq. 5.4.1, where d_{goal} is the distance from the robot to the goal, $d_{\text{coll.}} = 0.6$ is the distance between the centers of entities beneath which a collision is considered to have occurred, d_i is the distance between the robot and pedestrian i, $d_{\text{disc.}} = d_{\text{coll.}} + 0.2$ is the minimum "comfortable" distance between a robot and a pedestrian (as in [9]), and d_j is the distance from the robot to the edge of the convex hull surrounding group j:

$$\begin{aligned} R(t) = & C_{\text{prog.}}(d_{\text{goal}}(t-1) - d_{\text{goal}}(t)) \\ & + C_{\text{goal}}\delta(d_{\text{goal}}(t) < d_{\text{coll.}}) \\ & - C_{\text{disc.}}\sum_i (d_{\text{disc.}} - d_i(t))\delta(d_{\text{coll.}} \leq d_i(t) \leq d_{\text{disc.}}) \\ & - C_{\text{coll.}}\sum_i \delta(d_i(t) < d_{\text{coll.}}) \\ & - C_{\text{group}}\sum_j \delta(d_j(t) < d_{\text{coll.}}). \end{aligned} \tag{5.0}$$

The multiple objectives are weighted via the following constants: $C_{\text{prog.}} = 0.1$, $C_{\text{goal}} = 1.0$, $C_{\text{disc.}} = 0.5$, $C_{\text{coll.}} = 0.25$, and $C_{\text{group}} = 0.25$. The first term encourages the robot to progress toward the goal, allowing us to remove the initial imitation learning phase in [9]. The second, third, and fourth terms encourage the robot to reach the goal, avoid close encounters with pedestrians, and avoid collisions, respectively. The last term encourages the robot to adhere to group social norms by penalizing any incursion into a group's "space."

As seen in Fig. 5.2, to determine the d_j terms, I first compute a polygon representing the convex hull of the positions of all members of the pedestrian group. I then calculate the minimum distance between the robot and the polygon and penalize the robot for intruding into this space. Note that group membership information is only needed during the training phase and is not required for evaluation or deployment of the algorithm.

5.4.2 Neural Network Architecture

The overall network architecture is depicted in Fig. 5.3. As in [9], I used an attention-based network architecture to represent navigation policies. For each pedestrian, a vector of quantities representing the pedestrian was first concatenated with a vector representing the robot and passed through the first multi-layer perceptron (MLP) in the network (MLP_1). The resulting feature vector was concatenated with the mean value of the outputs of MLP_1 for all pedestrians and used to compute an attention score α_i for each pedestrian via MLP_3. The output of MLP_1 was also passed through MLP_2 to generate a separate “robot-pedestrian interaction vector” that was then multiplied by α_i to generate a weighted feature vector ω_i for each pedestrian. The ω_i were summed for all pedestrians and the result was passed through MLP$_4$, ultimately leading to separate policy and value heads. The filter parameters for the neural network architecture are described in Table 5.1. In summary, the architecture matched that of [9] without the interaction module and (1) with a softmax layer being added to produce a categorical policy output and (2) a single fully-connected layer with 100 neurons connecting to a scalar value head. This configuration enabled actor-critic learning, as described below.

5.4.3 PPO Algorithm

The agents were trained using proximal policy optimization (PPO; [174]), a leading model-free, actor-critic approach. Hyperparameters were chosen to mimic those used for Atari in [174], with the exceptions of shorter windows (16 steps) and more windows per batch (64). This change was made to accommodate the shorter episodes of CrowdNav while maintaining the number of experiences per batch.

Network Layer	**Output Features / Activation**
MLP_1	150, ReLU, 100, ReLU
MLP_2	100, ReLU, 50
MLP_3	100, ReLU, 100, ReLU, 1
MLP_4	150, ReLU, 100, ReLU, 100, ReLU
$Linear_1$	81
$Linear_2$	1

Table 5.1: Network Layer Filter Parameters

5.4.4 Training Details

I used the Adam optimizer [115] with learning rate set to 2.5e-4 and epsilon set to 1.0e-5. In the RL policy, the discount factor, γ, was set to 0.99 and the credit assignment variable, λ, was set to 0.95. I trained the policy for 7000 iterations yielding approximately 4.8M steps and reaching a maximum reward of approximately 1.7 based on the reward definition described in Eq. 5.4.1.

5.5 Experimental Evaluation

The goal of the experiment is to assess the efficacy of the group-aware navigation policy. Below, I describe the experimental setup, metrics, and results.

5.5.1 Experimental Setup

The experiments involved four settings determined by two factors: the number of pedestrians and the number of groups. I explored both 5-person and 10-

person settings as well as a single group and a stochastic number of groups as described in Sec. 5.4.

I used the Circle Crossing scenario where groups of pedestrians started and ended around the perimeter of a circle (radius = 4 m) during training and evaluation. The robot's starting and end positions were set to ensure the robot to go through the center of the circle and interact with the pedestrian groups. I evaluated the trained policy on 250 trials with randomly initialized starting and ending pedestrian positions for the four experimental settings. Lastly, the comparison baseline was based on [9], without inclusion of the group-aware reward term.

5.5.2 Metrics

The evaluation was focused on 1) robot navigation performance, 2) pedestrian navigation performance, and 3) social compliance. For robot navigation performance, the metrics represent the quality of the robot's ability to navigate to the goal quickly without collision. Specifically, I measured the following metrics:

- **Successes**: Number of trials the robot reached the target goal.
- **Collisions**: Number of trials the robot collided with a pedestrian.
- **Timeouts**: Number of trials the robot did not reach the goal within the

allotted time (25 seconds).

- **Time to Goal**: Average of the number of seconds the robot needed to reach the goal for all trials.
- **Mean Robot Velocity**: The average velocity of the robot at each time step during all trials.

To assess pedestrian performance, I measured the impact of the robot's navigation behavior on the desired pedestrian motion. Specifically, I measured:

- **Mean Pedestrian Velocity**: The average velocity of the pedestrians during the trials.
- **Mean Pedestrian Angle**: The average angular deviation between the pedestrian's observed motion and the direct vector to the pedestrian's goal. This metric seeks to measure the disturbance from the optimal trajectory to the goal caused by the robot's policy.

Finally, to assess social norms, I quantified how the robot maintained social distance among individual pedestrians and limited intersections with groups of pedestrians. For this, I considered the following metrics:

- **Group Intersections**: The number of groups intersections by the robot that occurred during the trials.

Method / #Grps / #Peds	**Successes ↑**	**Ped. Colls. ↓**	**Timeouts ↓**
Baseline / 1 / 5	**237**	11	**2**
Group Aware / 1 / 5	236	**9**	5
Baseline / 2.548 / 5	238	12	0
Group Aware / 2.548 / 5	**242**	**8**	0
Baseline / 1 / 10	222	23	5
Group Aware / 1 / 10	**232**	**14**	**4**
Baseline / 4.884 / 10	239	10	1
Group Aware / 4.884 / 10	**241**	**9**	**0**

Table 5.2: This table summarizes robot navigation performance including number of successes, collisions and timeouts across 5 and 10 pedestrians. Bold text indicates statistically significant results.

- **Individual Discomfort**: The mean distance between the robot and the pedestrians aggregated over all pedestrians when the robot violates the discomfort threshold (0.2 m).

- **Pedestrian Social Force**: The mean social force applied to pedestrian i. This is equal to the sum of the forces applied to pedestrian i from the other pedestrians and the robot, j as described in Sec. 5.3.4 (i.e., $\sum_j f_{ij}$). This metric captures how the robot's motion may impact directly or indirectly human pedestrians' motions.

- **Robot Social Force**: The mean social force applied to the robot, r from other pedestrians, j as described in Sec. 5.3.4 (i.e., $\sum_j f_{rj}$). This metric captures how the robot's motion may be impacted by human pedestrians.

Method / #Grps / #Peds	**Mean Time (s) ↓**		**Mean Robot Vel. (m/s) ↑**	
Baseline / 1 / 5	**8.24**	$t(471) = 9.62$	0.962	$t(498) = 0.60$
Group Aware / 1 / 5	8.92	$p < .001$	0.964	$p = .551$
Baseline / 2.548 / 5	**8.23**	$t(478) = 7.73$	0.964	$t(498) = 0.51$
Group Aware / 2.548 / 5	8.81	$p < .001$	0.961	$p = .610$
Baseline / 1 / 10	**8.59**	$t(452) = 11.19$	0.955	$t(498) = 0.21$
Group Aware / 1 / 10	9.87	$p < .001$	0.956	$p = .833$
Baseline / 4.884 / 10	**8.72**	$t(478) = 18.83$	**0.960**	$t(498) = 6.39$
Group Aware / 4.884 / 10	10.21	$p < .001$	0.918	$p < .001$

Table 5.3: This table summarizes the mean time to completion and the mean robot velocity across 5 and 10 pedestrians. Bold text indicates statistically significant results. I show that the group aware policy generally takes longer to reach the goal, which is expected because the robot navigates around the pedestrians. I also observe robot velocities are approximately the same.

Method / #Grps / #Peds	**Mean Ped. Vel. (m/s) ↑**		**Mean Ped. Angle (°) ↑**	
Baseline / 1 / 5	1.170	$t(498) = 5.18$	3.76	$t(498) = 1.65$
Group Aware / 1 / 5	**1.183**	$p < .001$	3.59	$p = .100$
Baseline / 2.548 / 5	1.136	$t(498) = 1.32$	5.99	$t(498) = 3.23$
Group Aware / 2.548 / 5	1.146	$p = .186$	**5.59**	$p = .001$
Baseline / 1 / 10	1.161	$t(498) = 2.06$	4.11	$t(498) = 1.86$
Group Aware / 1 / 10	**1.174**	$p = .040$	3.93	$p = .064$
Baseline / 4.884 / 10	1.089	$t(498) = 3.53$	8.09	$t(498) = 8.41$
Group Aware / 4.884 / 10	**1.108**	$p < .001$	**7.07**	$p < .001$

Table 5.4: This table summarizes the pedestrian velocity and disturbance angle from the desired goal across 5 and 10 pedestrians. Bold text indicates statistically significant results. I show that the group aware policy allows pedestrians to achieve faster velocity with less disturbance to the pedestrian's goal.

5.5.3 Results

An independent two-tailed t-test was conducted to compare the group-aware and the baseline policies. For all the statistical tests, an α level of .05 ($p < .05$)

Method / #Grps / #Peds	**Grp. Intersections ↓**	**Individual Discomfort ↓**	
Baseline / 1 / 5	143	3.10	$t(498) = 3.48$
Group Aware / 1 / 5	**15**	**1.29**	$p < .001$
Baseline / 2.548 / 5	151	2.87	$t(498) = 0.49$
Group Aware / 2.548 / 5	**22**	2.63	$p = .625$
Baseline / 1 / 10	176	4.20	$t(498) = 2.92$
Group Aware / 1 / 10	**29**	**2.31**	$p = .004$
Baseline / 4.884 / 10	258	4.94	$t(498) = 4.99$
Group Aware / 4.884 / 10	**20**	**2.29**	$p < .001$

Table 5.5: This table summarizes the number of instances a group was intersected by the robot and the individual discomfort observed by the pedestrians. Bold text indicates statistically significant results. Here, I show the group aware policy leads has significantly less number of group intersections and on overall reduced level of individual discomfort.

Method / #Grps / #Peds	**Ped. Social Force ↓**		**Robot Social Force ↓**	
Baseline / 1 / 5	0.375	$t(498) = 3.43$	0.523	$t(498) = 1.95$
Group Aware / 1 / 5	**0.351**	$p < .001$	0.482	$p = .051$
Baseline / 2.548 / 5	0.522	$t(498) = 3.95$	0.716	$t(498) = 2.78$
Group Aware / 2.548 / 5	**0.485**	$p < .001$	**0.657**	$p = .006$
Baseline / 1 / 10	0.395	$t(498) = 3.99$	0.707	$t(498) = 3.85$
Group Aware / 1 / 10	**0.366**	$p < .001$	**0.597**	$p < .001$
Baseline / 4.884 / 10	0.681	$t(498) = 7.32$	0.964	$t(498) = 4.80$
Group Aware / 4.884 / 10	**0.599**	$p < .001$	**0.849**	$p < .001$

Table 5.6: This table summarizes the pedestrian and robot social forces. Bold text indicates statistically significant results. Here, I show the group aware policy leads to reduced social force for both the pedestrian and the robot.

was used for significance.

Table 5.2 summarizes the robot navigation performance as well as their corresponding statistical test results. Overall, the group aware policy gener-

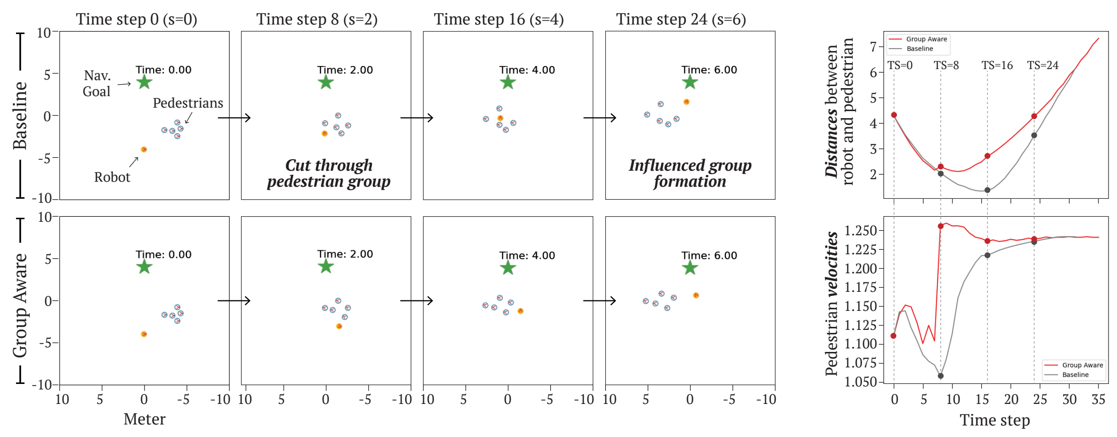


Figure 5.4: The figure on the left shows a representative example of the robot navigating through the crowd of pedestrians over time using both the baseline and the group aware policy. The baseline policy chooses actions that cut through the group of pedestrians and influences the group formation, while the group aware policy chooses actions that move around the group with minimal disturbance. The figure on the right shows the average distance between the pedestrian and the robot (top) and the average pedestrian velocity (bottom) over time. Here, I show the group aware policy results in increased distance to the pedestrians while allowing the pedestrians to maintain faster speeds.

ally led to higher number of successful trials, while minimizing the number of collisions and timeouts.

In Table 5.3, I compare the mean time of the robot to reach the goal and the average velocity. Here I show that the group-aware policy results in a more time needed to reach the target. This is to be expected as the robot prefers a policy that moves around the pedestrians versus driving through the groups. This leads to slightly deviated paths to the direct line to goal resulting in higher times as expected. It can also be seen that the group-aware policy does not lead to slower velocities as indicated by the results.

In Table 5.4, I evaluate the effect of the group-aware policy on the pedestrian's navigation. I measure average pedestrian velocity as well as the average angle deviation of the pedestrian from the optimal vector to the goal. The objective is to measure the relative disturbance of one policy to the other. Here, I can see the group-aware policy results in allowing the pedestrians to maintain a higher velocity while minimizing the overall disturbance to the goal.

In Table 5.5, I now compare the number of instances the robot chooses to navigate through the group versus not intersecting through it. I also compare the individual discomfort caused by navigating too close to the pedestrians. In both metrics, it can be seen that the group-aware policy outperforms the baseline. It results in an 88% improvement in reducing the number of instances where the robot navigated through a group. Moreover, the group aware policy resulted in a 43% reduction in individual discomfort.

Finally, in Table 5.6, I summarize the social compliance results with their corresponding statistical test results. As indicated by this table, I observe that the group-aware policy improved the overall social forces applied to the pedestrians and robot.

As mentioned above, the group-aware robot took a longer path to its goal due to its preference of navigating around groups of pedestrian to avoid group intersections, whereas the baseline robot aimed to reach its goal in spite of cutting in between groups of pedestrians. Fig. 5.4 illustrates an example of such behavior—the baseline policy chose a path that cut through the pedestrian group whereas, in the same scenario, the group-aware policy chose a path around the group. The resulted group-aware behavior ultimately enabled greater group cohesion and less disruption while improving group and individual discomfort. I additionally computed the distances between the pedestrians and the robot for both policies (Fig. 5.4 top-right) as well as the velocities of the pedestrians (Fig. 5.4 bottom-right) over time. During interaction between the robot and the pedestrians, I observe that the distance between the pedestrians and the robot was lower for the baseline policy corresponding to the results reported in Table 5.5. Further, we see that the average pedestrian velocity decreased substantially during the times of interaction in the baseline policy; however, we do not see similar decreases in the group-aware policy.

5.6 Additional Qualitative Examples

In Fig. 5.5 through 5.11, I show additional examples of the RL policy working with increased number of pedestrians with both 1 and a Poisson distributed group membership. Here I can see the affects of the group aware policy encourage the robot to move around the groups instead of disrupting the group trajectory.

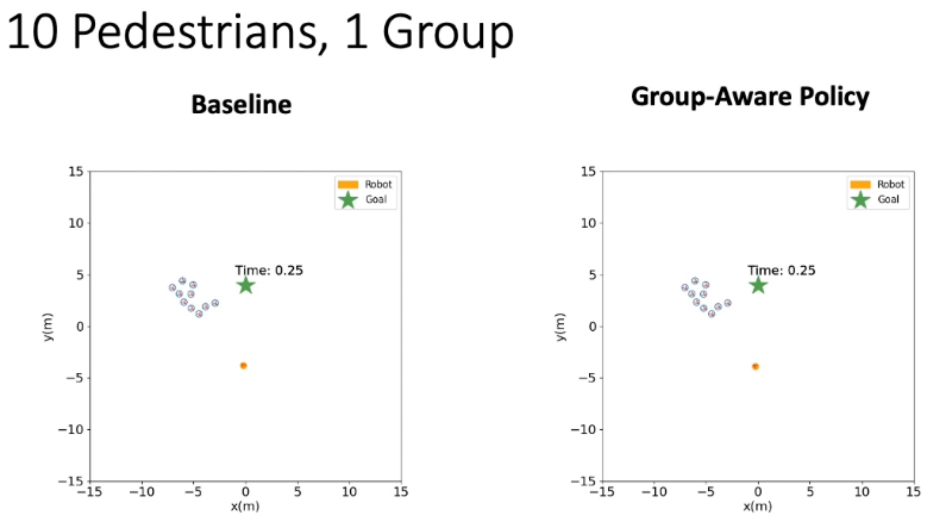


Figure 5.5: 10 Pedestrians, 1 Group, Start of Sequence

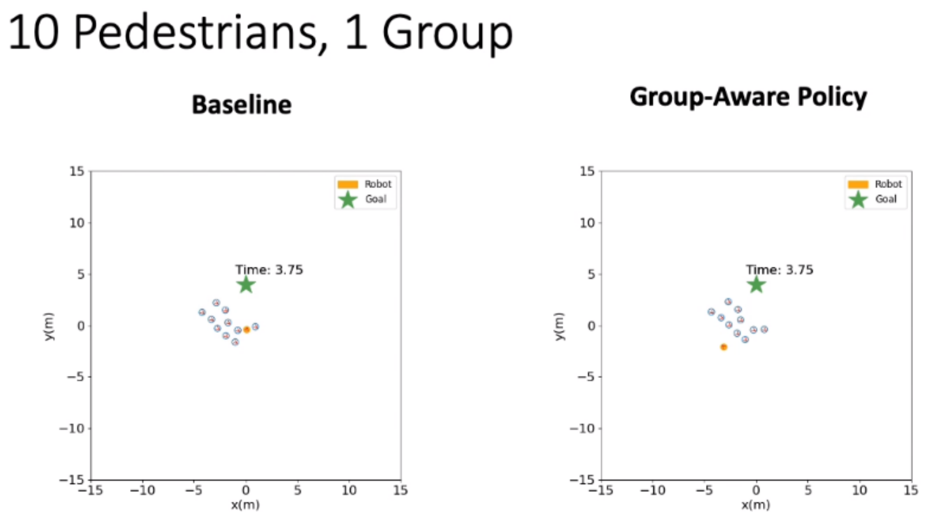


Figure 5.6: 10 Pedestrians, 1 Group, Middle of Sequence

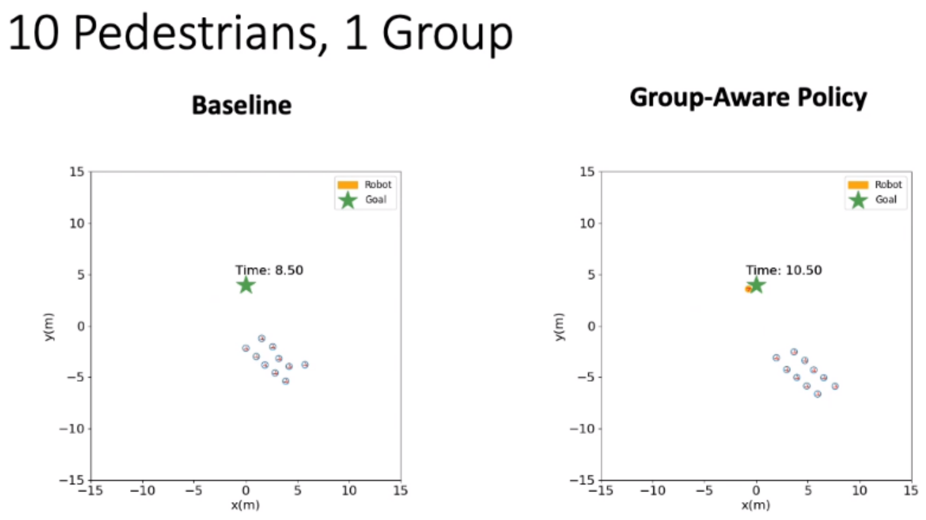


Figure 5.7: 10 Pedestrians, 1 Group, End of Sequence

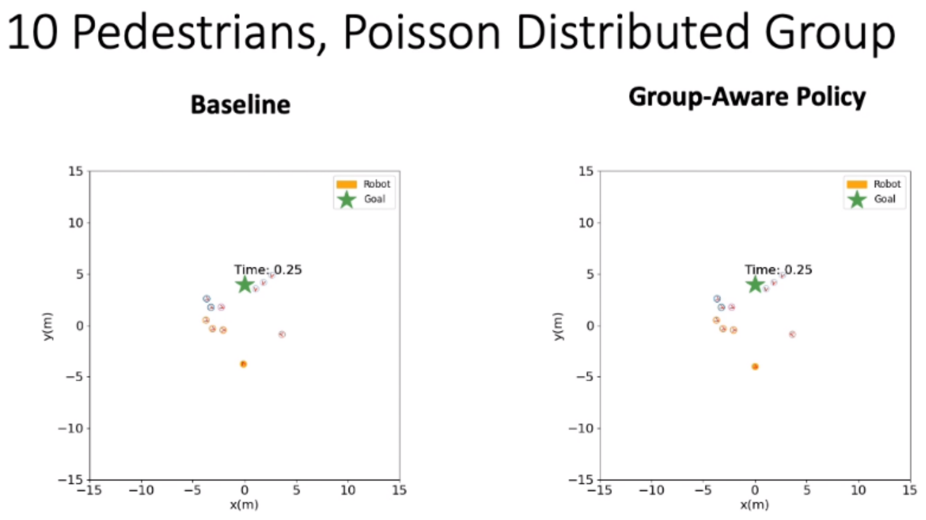


Figure 5.8: 10 Pedestrians, Poisson Distributed Group, Start of Sequence

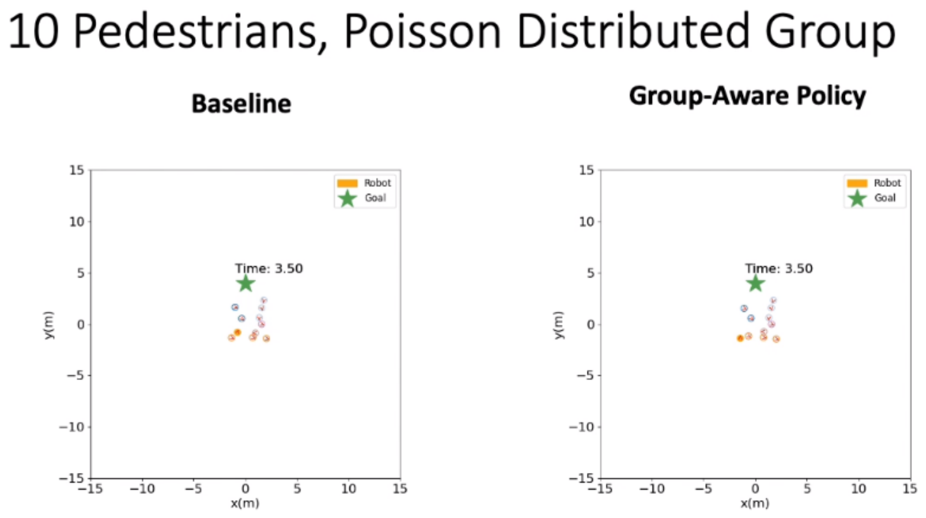


Figure 5.9: 10 Pedestrians, Poisson Distributed Group, Middle of Sequence

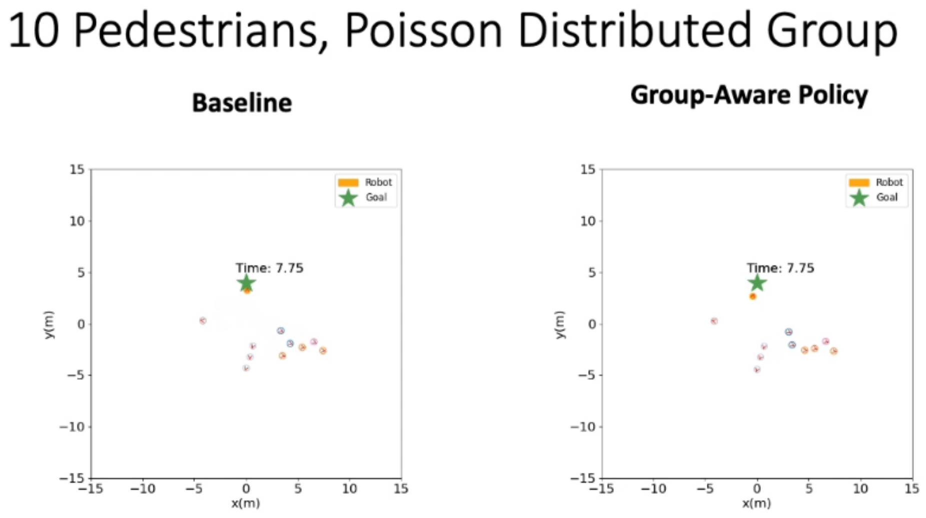


Figure 5.10: 10 Pedestrians, Poisson Distributed Group, End of Sequence

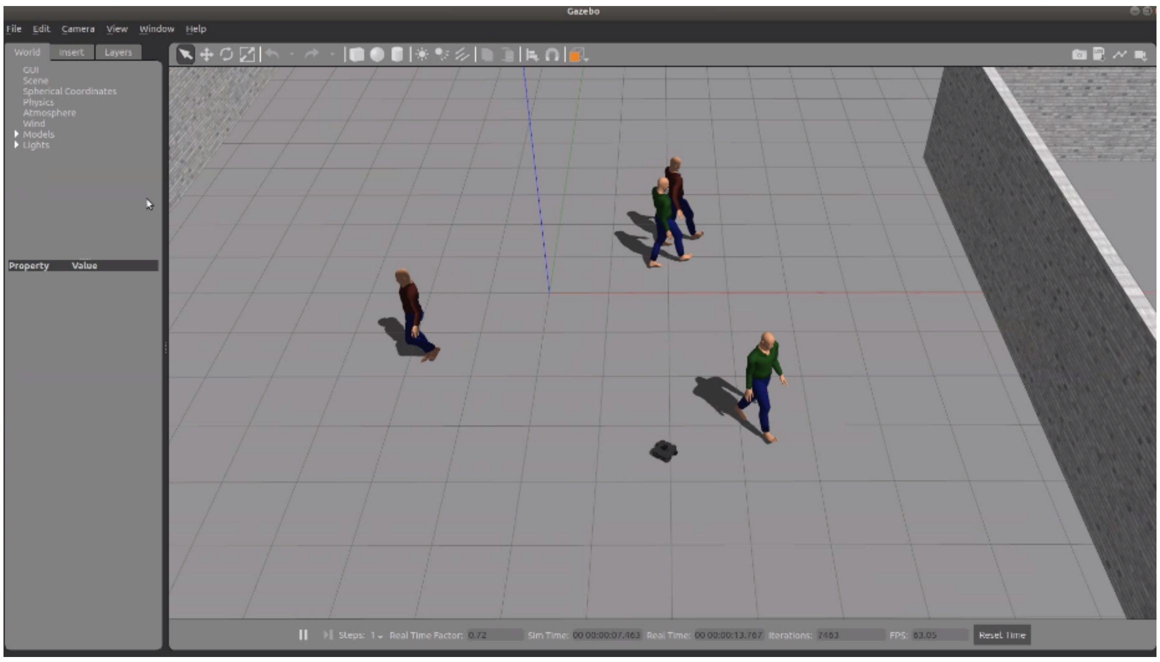


Figure 5.11: Gazebo Crowd Navigation

5.7 Experiments in Gazebo Simulation Environment

While the CrownNav simulation provides an ideal environment for training RL policies, it fails to model camera sensors capable of capturing perceptual data of the environment. To evaluate the efficacy of the learned policy on a more realistic simulation, I leverage the Gazebo simulator using pedestrian models and the Turtlebot as the mobile robot platform.

5.8 Discussion

Towards achieving socially appropriate robot navigation in human environments, this chapter explores group-aware behaviors that respect pedestrian group formations and trajectories while minimally sacrificing robot navigation performance. This approach utilizes deep reinforcement learning and considers group formation during training. The results show that the learned policy is able to achieve higher number of successful trials in which the robot reached the goal, fewer collisions, and less impact to the pedestrian's motion towards their goal. In addition, I show that the learned policy not only reduced the number of group violation (e.g., cutting through the group) but also decreased the individual discomfort and social forces applied to the pedestrians and robot. This approach, however, resulted in an increase of the robot's total time to goal compared to the baseline that did not consider group formation. This increase of robot navigation time was expected as the robot sought to move around groups as opposed to navigate through them (Fig. 5.4). However, the results show that even though the total time to goal increased, the average velocity of the robot was mostly unaffected by the group aware policy.

This exploration indicates several directions of future research. First, I would like to determine how well the learned policy reflects actual human motion through groups of pedestrians. Second, I would like to investigate whether

I can bootstrap the learned policy with imitation learning using observations of humans navigating groups of pedestrians. Third, I would like to investigate different representations of group space beyond the convex hull approach described in this chapter. These additional parameters include social interaction during movement, the specific formation of the group, and environmental cues (e.g., social space) that may contribute to learning more socially compliant navigation policies. Additionally, while this chapter focuses mostly on robot and pedestrian navigation performance as well as group intersections and discomfort, there are other factors to consider for socially appropriate behavior such as how to pass and follow human groups. Finally, while learning a single policy works for limited number of environments, choosing amongst a library of policies depending on the density of pedestrians, the layout of the environment, and the local culture can lead to better navigation performance once deployed. For example, respecting human grouping may not always be possible (e.g., in a narrow corridor). It is therefore important for a mobile robot to selectively choose a context suitable policy in order to achieve efficient, safe, and socially appropriate navigation in crowded human environments.

Chapter 6

Conclusions and Future Work

In this thesis, I established the foundation to integrate perception, prediction and uncertainty to improve navigation performance in environments with static and dynamic obstacles including pedestrians. By introducing concepts such as occupancy map prediction with uncertainty, intent-aware pedestrian prediction with uncertainty and group-aware navigation, navigation performance was improved along key metrics including improving the exploration efficiency, increasing the number of successful trials, reducing the number of collisions and timeouts and improving the overall social compliance of the navigation policy.

In Chapter 3, I developed techniques that use generative neural networks to make predictions of future occupied spaces. I assume the ability to use depth and camera data to generate an occupancy map that is limited by the field of view of the sensor. Using semi-supervised strategies, I develop the ability to generate occupancy maps that extend the line of sight of the sensor that are subsequently used by exploration and control algorithms. I first investigate various generative neural network architectures and optimization strategies to determine optimal architectures and learning strategies that produce predicted occupancy maps with the highest accuracy. Multiple hypothesis loss techniques were then used to produce an environmental uncertainty metric. By leveraging this metric, the efficiency of the exploration strategy was improved compared to state-of-the-art exploration approaches. In addition, a controller that uses the predicted regions to enable high speed navigation was developed. Here, the ability to travel at higher speeds with fewer collisions was demonstrated on a physical small platform race car.

In Chapter 4, I focused on navigation in the presence of pedestrians. I first develop techniques that perform state and uncertainty estimation of pedestrians using high-dimensional observations as the input. By using traditional Kalman Filter techniques on latent embeddings representing the pedestrian, an improvement to the state and uncertainty estimation was shown while reducing the computational requirements. I then studied predicting future motion of pedestrians. I showed that by reasoning about the pedestrian's intent, the prediction accuracy can be improved on standard datasets. I then develop an adaptive reinforcement learning algorithm that leverages errors in predicted pedestrian motion as an uncertainty metric to improve navigation performance while reducing the number of collisions.

Finally, in Chapter 5, I extended the RL approaches developed in Chapter 4 to consider navigation through groups of pedestrians. I show that by considering group membership as part of the RL policy, key navigation performance metrics including number of successful trials, reduced number of collisions, timeouts can be significantly improved. In addition, improved performance with respect to pedestrian navigation preferences including reducing the number of group intersections and individual discomfort, while minimizing the disturbance to the pedestrian from navigating to their goals was also demonstrated.

6.1 Limitations

There are a number of limitations related to the ideas presented in this thesis that provide opportunities for future research. In Chapter 3, much of the map prediction work was trained and evaluated in indoor, office-like environments. While I believe these approaches will translate to outdoor environments, it is an area of future evaluation. Further, the existing approach still requires an explicit map to be generated which is known to be computationally expensive to produce. This could potentially limit the ability to travel at high speeds and is an area to investigate. Finally, the use of a multi-headed network to produce multiple-hypotheses generates a fixed number of hypotheses during inference. This fixed number may be difficult to tune and would likely change depending on the complexity of the environment. Additional work towards using variational autoencoders in the latent dimension could help address this limitation.

There are also a number of improvements that can be made with respect to navigating in the presence of pedestrians. First, the average navigation time of the learned policies did increase compared to the other approaches. Although, this was primarily caused by switching to a risk-averse policy or selecting a suboptimal trajectory to improve social compliance, additional work could be focused on improving the time to goal metric. Second, while the number of collisions have decreased through the learned policy, they have not been eliminated entirely. Addressing this would involve improving intent estimation, better prediction algorithms, and investigating new approaches to uncertainty estimation.

Another limitation is in the approach to pedestrian intent modeling. The current work focuses on estimating the goal of the pedestrian as the intent. However, there are many latent variables that could affect the pedestrian's navigation preference that could be expanded upon. Additionally, I focus on a fairly limited scenario of circle crossing in many of the simulation testing. This was mainly due to the prevalence of existing literature using this scenario for evaluation. Future work should consider additional and more complex environments and scenarios. Finally, with group navigation, I primarily focus on group intersection and individual discomfort for assessing social normal behavior. There are other factors that could be considered including preferable navigation strategies depending on group formation, how to pass a group of individuals, as well as how to minimize disturbance to the group in scenarios where crossing the group is strictly necessary.

6.2 Future Work

In the previous section, I describe several limitations of the work presented in this thesis. In this section, I review several ongoing and future efforts to continue this line of research to address the limitations described above.

6.2.1 Map Prediction and High Speed Navigation in Off-Road Rough Terrain Environments

In this thesis, I have demonstrated the ability to learn spatial structures in indoor environments that act as predictors for future occupied spaces. We

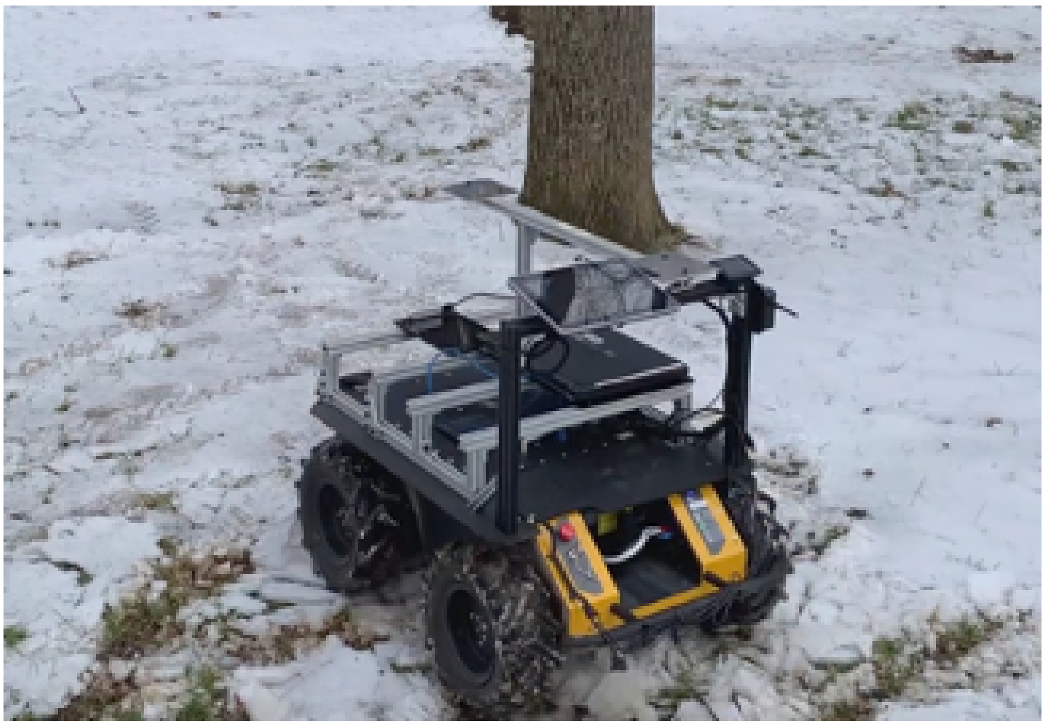


Figure 6.1: Outdoor experiments of map prediction and navigation using the Clearpath Husky Robot.

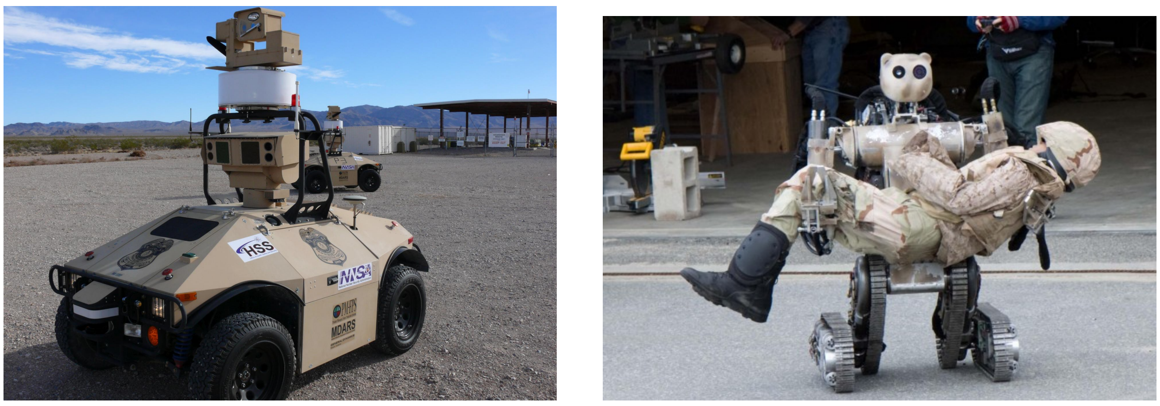

(a) Border Patrol Robot **(b)** Combat Casualty Robot

Figure 6.2: Examples of robots needed for military and civilian operations.

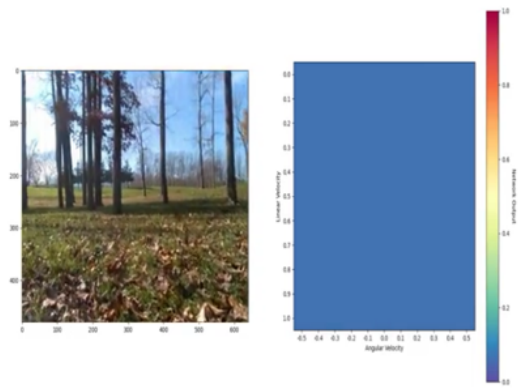
are currently evaluating the applicability of this approach in outdoor, off-road, rough terrain environments. This is a key interest area in particular to military sponsors with applications ranging from border patrol (Fig. 6.2a), combat casualty care (Fig. 6.2b), and warfighter support and protection. A key capability needed is to support high-speed maneuvers reliably on rough terrains.

For this reason, we are translating many of the research ideas developed as part of this thesis to outdoor navigation. In Fig. 6.1, I show experimentation using the Clearpath Husky robot equipped with an Intel RealSense camera used for depth sensing and odometry. We are leveraging the same mapping algorithm used to produce occupancy maps similar to the approach taken in Chapter 3. The objective is to learn spatial features in outdoor environments for future prediction and subsequently leverage the predicted regions for efficient exploration and high speed navigation as presented in Chapter 3 in indoor environments.

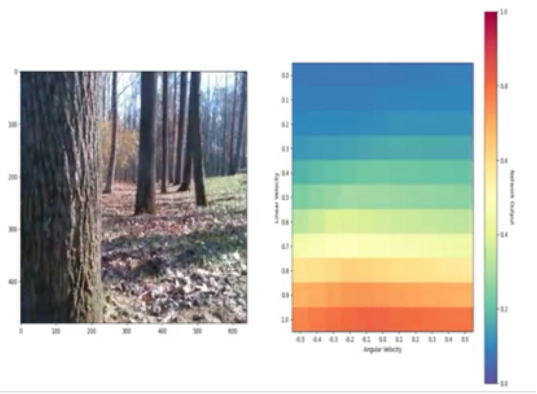
In addition, we will continue to improve upon techniques to better estimate uncertainty by generating multiple hypotheses that accurately represent the true distribution of predicted futures. This will be accomplished by adding a variational latent distribution as part of the encoder/decoder network. The learned variational latent distribution will allow us to sample and generate multiple hypotheses. We believe this approach will integrate well with the controller described in Chapter 3 by generating a distribution of trajectories as well as to develop risk-sensitive control algorithms based on the variance of the multiple hypotheses.

6.2.2 Predictive Traversability

We also extended the work presented in this thesis to predict traversable regions in the space. The premise is fundamentally similar to map prediction. Given a high resolution input such as raw camera observations, we seek the answer to: Can we predict areas that are traverasable conditioned by actions? We formulate this problem by regressing the trajectory tracking error. Formally, given an image and action sequence, we estimate the probability that the actions will be executed correctly as determined by predicting trajectory tracking error. We have conducted preliminary experiments in outdoor environments to assess the feasibility of this approach. In a semi-supervised manner, a Clearpath Huskey robot navigates a wooded environment (Fig. 6.1) at



(a) Traversability Prediction without Obstacles



(b) [Traversability Prediction with Obstacles

Figure 6.3: This figure shows the output of the neural network that learns a probability of trajectory error given the input image conditioned by the actions, angular velocity (x-axis) and linear velocity (y-axis). One the left, we show the output of the neural network that correctly predicts low trajectory error for all linear and angular velocity combinations. On the right, we show that high linear velocities regardless of the angular velocity would lead to high trajectory errors caused by collision. For lower velocities, the neural network predicts angular velocities turning away from the tree will produce lower trajectory errors which is ultimately selected by the planning algorithm.

varying linear and angular velocities. We developed a neural network that receives an RGB image from the robot's perspective and potential control actions and produces a probabilistic map that predicts the likelihood of an error between the expected trajectory and the observed trajectory. We have shown that this is effective at detecting obstacles as seen in Fig. 6.3. In Fig. 6.3a, we can see that since there are no obstacles in the immediate horizon, all linear and angular velocities result in low expected trajectory error. In Fig. 6.3b, the network detects the tree obstacle and favors slower velocities as well as angular velocities that drive the robot towards the right away from the obstacle.

There are many ways to extend this to support high speed maneuvers in complex environments. The algorithms must not only perform static obstacle avoidance but also choose paths that contain smoother terrains, reason about what obstacles will present navigation challenges as well as avoid negative obstacles in the environment. These additional capabilities can also be formulated similar to obstacle avoidance using predicted trajectory error. For example, driving through rough terrain will lead to a mismatch between observed and predicted trajectories. By predicting this error, we can select trajectories that favor smoother terrains. The ongoing research will focus on developing optimal neural network architectures that are capable of reasoning about more complex artifacts of the environment needed for high speed navigation.

6.2.3 Improved Social Compliant Navigation

While the ideas presented in Chapters 4 and 5 make good progress towards developing techniques for social navigation, many improvements can be made to increase the navigation and social compliance performance. As described in Sec. 6.1, I primarily focus on estimating intent uniformly throughout the environment. While this was shown to be effective, it doesn't capture contextual and semantic cues that could further improve intent estimation. The rationale behind this approach is that certain areas in the environment may be more traversable than others and could be used to improve intent estimation. Further, contextual cues in the environment maybe indicate where the pedestrian may be headed. An example is that an individual may visit the water fountain after the restroom. Using semantic information from the scene can allow us to reason about spatial relationships that can be used for better prediction.

Another area to explore is the use of inverse reinforcement learning methods to learn socially compliant behavior. Often, socially compliant behavior is difficult to formalize in an equation used as the RL reward function. In addition, weighting different trade-offs such as reward for reaching the goal versus penalties associated with social discomfort can be difficult to tune and heuristically determine. An area we are actively exploring is the use of inverse RL to learn the reward function by observing humans in a variety of different contexts and environments. The goal is to use the learned reward function to train an RL policy to match the human behavior. The hope is that this will lead to even more socially compliant navigation policies that generalize better to new situations and environments.

Bibliography

[1] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an image-based motion context for multiple people tracking," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3542–3549.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, July 1997, pp. 146–151.

[4] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with bayesian optimization," in *Intelligent Robots and Systems*

(IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016, pp. 1816–1822.

[5] S. Bai, J. Wang, K. Doherty, and B. Englot, “Inference-enabled information-theoretic exploration of continuous action spaces,” in *International Symposium of Robotics Research*, 2015.

[6] S. Wirth and J. Pellenz, “Exploration transform: A stable exploring algorithm for robots in rescue environments,” in *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on.* IEEE, 2007, pp. 1–5.

[7] L. Bertoni, S. Kreiss, and A. Alahi, “Monoloco: Monocular 3d pedestrian localization and uncertainty estimation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal dataset for autonomous driving,” *arXiv preprint arXiv:1903.11027*, 2019.

[9] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning,” *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6015–6022, 2019.

[10] R. R. Murphy, *Disaster Robotics*. The MIT Press, 2014.

[11] J. Moore, K. C. Wolfe, M. S. Johannes, K. D. Katyal, M. P. Para, R. J. Murphy, J. Hatch, C. J. Taylor, R. J. Bamberger, and E. Tunstel, "Nested marsupial robotic system for search and sampling in increasingly constrained environments," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2016, pp. 002 279–002 286.

[12] K. D. Katyal, C. Y. Brown, S. A. Hechtman, M. P. Para, T. G. McGee, K. C. Wolfe, R. J. Murphy, M. D. Kutzer, E. W. Tunstel, M. P. McLoughlin *et al.*, "Approaches to robotic teleoperation in a disaster scenario: From supervised autonomy to direct control," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1874–1881.

[13] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, May 2005.

[14] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, p. 1309–1332, 2016.

[15] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a sur-

vey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, 12 2017.

[16] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[17] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[18] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[19] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: socially acceptable trajectories with generative adversarial networks," *CoRR*, vol. abs/1803.10892, 2018. [Online]. Available: http://arxiv.org/abs/1803.10892

[20] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, and S. Savarese, "Sophie: An attentive GAN for predicting paths compliant to social and physical constraints," *CoRR*, vol. abs/1806.01482, 2018. [Online]. Available: http://arxiv.org/abs/1806.01482

[21] F. Bartoli, G. Lisanti, L. Ballan, and A. D. Bimbo, “Context-aware trajectory prediction,” *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 1941–1946, 2017.

[22] A. Vemula, K. Mülling, and J. Oh, “Modeling cooperative navigation in dense human crowds,” *CoRR*, vol. abs/1705.06201, 2017. [Online]. Available: http://arxiv.org/abs/1705.06201

[23] S. Pellegrini, A. Ess, and L. Van Gool, “Improving data association by joint modeling of pedestrian trajectories and groupings,” in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 452–465.

[24] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” 2020.

[25] D. P. Kingma and M. Welling, “An introduction to variational autoencoders,” *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through

deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1889–1897. [Online]. Available: http://proceedings.mlr.press/v37/schulman15.html

[28] 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

[29] H. Voicu and N. Schmajuk, "Exploration, navigation and cognitive mapping," *Adaptive Behavior*, vol. 8, no. 3-4, pp. 207–223, 2000. [Online]. Available: https://doi.org/10.1177/105971230000800301

[30] R. L. Buckner, "The role of the hippocampus in prediction and imagination," *Annual review of psychology*, vol. 61, pp. 27–48, 2010.

[31] P. Schwartenbeck, J. Passecker, T. Hauser, T. H. B. FitzGerald, M. Kronbichler, and K. J. Friston, "Computational mechanisms of curiosity and goal-directed exploration," *bioRxiv*, 2018. [Online]. Available: https://www.biorxiv.org/content/early/2018/09/07/411272

[32] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[33] T. F. Chan and J. Shen, “Mathematical models for local nontexture inpaintings,” *SIAM J. Appl. Math*, vol. 62, pp. 1019–1043, 2002.

[34] R. A. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. N. Do, “Semantic image inpainting with perceptual and contextual losses,” *CoRR*, vol. abs/1607.07539, 2016. [Online]. Available: http://arxiv.org/abs/1607.07539

[35] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2786–2793.

[36] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” *arXiv preprint arXiv:1603.02199*, 2016.

[37] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, “Action-Conditional Video Prediction using Deep Networks in Atari Games,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence,

D. D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2845–2853.

[38] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, “Deep image prior,” *CoRR*, vol. abs/1711.10925, 2017. [Online]. Available: http://arxiv.org/abs/1711.10925

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[40] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *CVPR*, 2017.

[41] S. T. O’Callaghan and F. T. Ramos, “Gaussian process occupancy maps,” *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012. [Online]. Available: https://doi.org/10.1177/0278364911421039

[42] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, “Value iteration networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2154–2162.

[43] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," *CoRR*, vol. abs/1703.00420, 2017. [Online]. Available: http://arxiv.org/abs/1703.00420

[44] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," *CoRR*, vol. abs/1709.10489, 2017. [Online]. Available: http://arxiv.org/abs/1709.10489

[45] G. J. Stein, C. Bradley, and N. Roy, "Learning over subgoals for efficient navigation of structured, unknown environments," in *Conference on Robot Learning*, 2018, pp. 213–222.

[46] C. Richter, J. Ware, and N. Roy, "High-speed autonomous navigation of unknown environments using learned probabilities of collision," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6114–6121.

[47] O. Asraf and V. Indelman, "Experience-based prediction of unknown environments for enhanced belief space planning," 10 2020.

[48] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone, "Map-Predictive Motion Planning in Unknown Environments," in *Proceedings - IEEE Inter-*

national Conference on Robotics and Automation. Institute of Electrical and Electronics Engineers Inc., may 2020, pp. 8552–8558.

[49] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *CoRR*, vol. abs/1702.01182, 2017. [Online]. Available: http://arxiv.org/abs/1702.01182

[50] P. Karkus, D. Hsu, and W. S. Lee, "Qmdp-net: Deep learning for planning under partial observability," in *Advances in Neural Information Processing Systems*, 2017, pp. 4694–4704.

[51] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE transactions on neural networks and learning systems*, 2019.

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

[53] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, March 2017.

[54] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time

style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016. [Online]. Available: http://arxiv.org/abs/1603.08155

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[56] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, 2017. [Online]. Available: http://arxiv.org/abs/1703.10593

[57] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. [Online]. Available: http://octomap.github.com

[58] P. Min, "binvox," `http://www.patrickmin.com/binvox`, 2004 - 2017, accessed: 2017-02-20.

[59] F. S. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, 2003.

[60] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin,

A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[61] J. R. Fienup, “Invariant error metrics for image reconstruction,” *Appl. Opt.*, vol. 36, no. 32, pp. 8352–8357, Nov 1997. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-36-32-8352

[62] C. Rupprecht, I. Laina, M. Baust, F. Tombari, G. D. Hager, and N. Navab, “Learning in an uncertain world: Representing ambiguity through multiple hypotheses,” *CoRR*, vol. abs/1612.00197, 2016. [Online]. Available: http://arxiv.org/abs/1612.00197

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

[64] B. Cain, “High speed autonomous vehicles using the mit racecar platform,” 2017.

[65] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system.” [Online]. Available: https://www.ros.org

[66] C. Xie, S. Liu, C. Li, M. Cheng, W. Zuo, X. Liu, S. Wen, and E. Ding, “Image inpainting with learnable bidirectional attention maps,” in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019,*

Seoul, Korea (South), October 27 - November 2, 2019. IEEE, 2019, pp. 8857–8866.

[67] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11218. Springer, 2018, pp. 3–19.

[68] K. D. Katyal, K. M. Popek, C. Paxton, J. L. Moore, K. C. Wolfe, P. Burlina, and G. D. Hager, "Occupancy map prediction using generative and fully convolutional networks for vehicle navigation," *CoRR*, vol. abs/1803.02007, 2018. [Online]. Available: http://arxiv.org/abs/1803.02007

[69] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019.

[70] T. Kaneko and T. Harada, "Noise robust generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[71] M. Basescu and J. Moore, "Direct nmpc for post-stall motion planning with fixed-wing uavs," *arXiv preprint arXiv:2001.11478*, 2020.

[72] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[73] K. Yang and S. Sukkarieh, “An analytical continuous-curvature path-smoothing algorithm,” *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[74] G. Kahn, P. Abbeel, and S. Levine, “BADGR: an autonomous self-supervised learning-based navigation system,” *CoRR*, vol. abs/2002.05700, 2020. [Online]. Available: https://arxiv.org/abs/2002.05700

[75] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. [Online]. Available: http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance

[76] C. Montella, “The kalman filter and related algorithms: A literature review,” 05 2011.

[77] S. Y. Chen, “Kalman filter for robot vision: A survey,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

[78] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, “Backprop kf: Learning discriminative deterministic state estimators,” in *Advances in Neural*

Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4376–4384. [Online]. Available: http://papers.nips.cc/paper/6090-backprop-kf-learning-discriminative-deterministic-state-estimators.pdf

[79] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors," 2018.

[80] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," 2015.

[81] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," 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. 3601–3610.

[82] P. Becker, H. Pandya, G. H. W. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann, "Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces," *CoRR*, vol. abs/1905.07357, 2019. [Online]. Available: http://arxiv.org/abs/1905.07357

[83] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," 04 2017.

[84] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham, "Dynanet: Neural kalman dynamical model for motion estimation and prediction," 2020.

[85] D. M. Richard and P. R. Lippmann, "Neural network classifiers estimate bayesian a posteriori probabilities," *Neural Computation*, pp. 461–483, 1991.

[86] R. M. Neal, *Bayesian Learning for Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 1996.

[87] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" 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. 5574–5584.

[88] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. [Online]. Available: http://proceedings.mlr.press/v48/gal16.html

[89] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dil-

lon, B. Lakshminarayanan, and J. Snoek, “Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 13 991–14 002.

[90] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *European Conference on Computer Vision (ECCV)*, September 2014.

[91] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[92] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, “Human-aware robot navigation: A survey,” *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[93] J. P. van den Berg, M. C. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” *2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935, 2008.

[94] J. van den Berg, S. Guy, M. Lin, and D. Manocha, *Reciprocal n-Body Collision Avoidance*, 04 2011, vol. 70, pp. 3–19.

[95] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," 06 2011, pp. 5628 – 5635.

[96] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, Jul 2013. [Online]. Available: https://doi.org/10.1007/s10514-013-9334-3

[97] 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. [Online]. Available: https://doi.org/10.1177/0278364915619772

[98] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Sep. 2018. [Online]. Available: https://arxiv.org/pdf/1805.01956.pdf

[99] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," 2018.

[100] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-

aware reinforcement learning for collision avoidance," *ArXiv*, vol. abs/1702.01182, 2017.

[101] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2543–2549.

[102] A. Elnagar, "Prediction of moving objects in dynamic environments using kalman filters," in *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*, July 2001, pp. 414–419.

[103] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *2008 IEEE Intelligent Vehicles Symposium*, June 2008, pp. 1068–1073.

[104] T. Batz, K. Watson, and J. Beyerer, "Recognition of dangerous situations within a cooperative group of vehicles," in *2009 IEEE Intelligent Vehicles Symposium*, June 2009, pp. 907–912.

[105] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[106] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 464–469.

[107] H. Xue, D. Q. Huynh, and M. Reynolds, "Bi-prediction: Pedestrian trajectory prediction based on bidirectional lstm classification," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov 2017, pp. 1–8.

[108] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *ArXiv*, vol. abs/1905.06113, 2019.

[109] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 4075–4084. [Online]. Available: http://proceedings.mlr.press/v80/pearce18a.html

[110] A. Khosravi, S. Nahavandi, and D. Creighton, "A prediction interval-based approach to determine optimal structures of neural network meta-

models," *Expert systems with applications*, vol. 37, no. 3, pp. 2377–2387, 2010.

[111] D. Su, Y. Y. Ting, and J. Ansel, "Tight prediction intervals using expanded interval minimization," *CoRR*, vol. abs/1806.11222, 2018. [Online]. Available: http://arxiv.org/abs/1806.11222

[112] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," *CVPR*, pp. 11 977–11 986, 2019.

[113] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 1, 1994, pp. 55–60 vol.1.

[114] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[115] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization,"

in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[116] S. Qi and S. Zhu, “Intent-aware multi-agent reinforcement learning,” *CoRR*, vol. abs/1803.02018, 2018. [Online]. Available: http://arxiv.org/abs/1803.02018

[117] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[118] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. K. Chandraker, “DESIRE: distant future prediction in dynamic scenes with interacting agents,” *CoRR*, vol. abs/1704.04394, 2017. [Online]. Available: http://arxiv.org/abs/1704.04394

[119] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[120] Y. F. Chen, M. Liu, M. Everett, and J. How, “Decentralized non-

communicating multiagent collision avoidance with deep reinforcement learning," 05 2017, pp. 285–292.

[121] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390681.1442794

[122] B. Efron, *The Jackknife, the bootstrap and other resampling plans*, ser. CBMS-NSF Reg. Conf. Ser. Appl. Math. Philadelphia, PA: SIAM, 1982, lectures given at Bowling Green State Univ., June 1980. [Online]. Available: https://cds.cern.ch/record/98913

[123] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[124] C. Rupprecht, I. Laina, R. Dipietro, M. Baust, F. Tombari, N. Navab, and G. Hager, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *Proceedings - 2017 IEEE International Conference on Computer Vision, ICCV 2017*, vol. 2017-October. Institute of Electrical and Electronics Engineers Inc., 12 2017, pp. 3611–3620.

[125] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager,

"Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5453–5459.

[126] A. Bera, T. Randhavane, and D. Manocha, "Aggressive, tense or shy? identifying personality traits from crowd videos." in *IJCAI*, 2017, pp. 112–118.

[127] K. Charalampous, I. Kostavelis, and A. Gasteratos, "Recent trends in social aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 93, pp. 85–104, 2017.

[128] A. Pandey, S. Pandey, and D. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review," *Int Rob Auto J*, vol. 2, no. 3, p. 00022, 2017.

[129] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 137–153, 2015.

[130] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[131] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation

in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[132] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European conference on computer vision*. Springer, 2016, pp. 549–565.

[133] A. Bera, T. Randhavane, R. Prinja, and D. Manocha, "Sociosense: Robot navigation amongst pedestrians with social and psychological constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 7018–7025.

[134] R. Mead and M. J. Matarić, "Autonomous human–robot proxemics: socially aware navigation based on interaction potential," *Autonomous Robots*, vol. 41, no. 5, pp. 1189–1201, 2017.

[135] M. Costa, "Interpersonal distances in group walking," *Journal of Nonverbal Behavior*, vol. 34, no. 1, pp. 15–26, 2010.

[136] A. F. Aveni, "The not-so-lonely crowd: Friendship groups in collective behavior," *Sociometry*, pp. 96–99, 1977.

[137] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The

walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PloS one*, vol. 5, no. 4, p. e10047, 2010.

[138] B. Okal and K. O. Arras, "Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2889–2895.

[139] Z. Yücel, F. Zanlungo, T. Ikeda, T. Miyashita, and N. Hagita, "Deciphering the crowd: Modeling and identification of pedestrian group motion," *Sensors*, vol. 13, no. 1, pp. 875–897, 2013.

[140] N. Bisagno, B. Zhang, and N. Conci, "Group lstm: Group trajectory prediction in crowded scenarios," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 0–0.

[141] 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.

[142] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, "Porca: Modeling and planning for autonomous driving among many pedestrians," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418–3425, 2018.

[143] K. D. Katyal, G. D. Hager, and C.-M. Huang, "Intent-aware pedestrian prediction for adaptive crowd navigation," in *2020 IEEE International*

Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 3277–3283.

[144] M. Luber, L. Spinello, J. Silva, and K. O. Arras, “Socially-aware robot navigation: A learning approach,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 902–907.

[145] M. Shiomi, F. Zanlungo, K. Hayashi, and T. Kanda, “Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model,” *International Journal of Social Robotics*, vol. 6, no. 3, pp. 443–455, 2014.

[146] M. Sebastian, S. B. Banisetty, and D. Feil-Seifer, “Socially-aware navigation planner using models of human-human interaction,” in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017, pp. 405–410.

[147] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, “You’ll never walk alone: Modeling social behavior for multi-target tracking,” in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 261–268.

[148] C. Johnson and B. Kuipers, “Socially-aware navigation using topological maps and social norm learning,” in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 151–157.

[149] R. Kirby, R. Simmons, and J. Forlizzi, "Companion: A constraint-optimizing method for person-acceptable navigation," in *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2009, pp. 607–612.

[150] X.-T. Truong and T.-D. Ngo, "Dynamic social zone based mobile robot navigation for human comfortable safety in social environments," *International Journal of Social Robotics*, vol. 8, no. 5, pp. 663–684, 2016.

[151] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[152] E. T. Hall, *The hidden dimension*. Garden City, NY: Doubleday, 1966, vol. 609.

[153] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, and D. Prattichizzo, "Walking ahead: The headed social force model," *PloS one*, vol. 12, no. 1, p. e0169734, 2017.

[154] G. Ferrer and A. Sanfeliu, "Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1730–1735.

[155] P. Ratsamee, Y. Mae, K. Ohara, T. Takubo, and T. Arai, "Human–

robot collision avoidance using a modified social force model with body pose and face orientation," *International Journal of Humanoid Robotics*, vol. 10, no. 01, p. 1350008, 2013.

[156] M. Swofford, J. C. Peruzzi, N. Tsoi, S. Thompson, R. Martín-Martín, S. Savarese, and M. Vázquez, "Improving social awareness through dante: A deep affinity network for clustering conversational interactants," *arXiv preprint arXiv:1907.12910*, 2019.

[157] Y. Kato, T. Kanda, and H. Ishiguro, "May i help you?-design of human-like polite approaching behavior," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2015, pp. 35–42.

[158] J. V. Gómez, N. Mavridis, and S. Garrido, "Fast marching solution for the social path planning problem," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1871–1876.

[159] A. Taylor, D. M. Chan, and L. D. Riek, "Robot-centric perception of human groups," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 3, pp. 1–21, 2020.

[160] M. Luber and K. O. Arras, "Multi-hypothesis social grouping and tracking for mobile robots." in *Robotics: Science and Systems*, 2013.

[161] J. Shao, C. Change Loy, and X. Wang, "Scene-independent group profiling

in crowd," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2219–2226.

[162] V. V. Unhelkar, C. Pérez-D'Arpino, L. Stirling, and J. A. Shah, "Human-robot co-navigation using anticipatory indicators of human walking motion," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 6183–6190.

[163] F. Yang and C. Peters, "Social-aware navigation in crowds with static and dynamic groups," in *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, 2019, pp. 1–4.

[164] Y. Du, N. J. Hetherington, C. L. Oon, W. P. Chan, C. P. Quintero, E. Croft, and H. M. Van der Loos, "Group surfing: A pedestrian-based approach to sidewalk robot navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6518–6524.

[165] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[166] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks,"

in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[167] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[168] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," *arXiv preprint arXiv:1909.13165*, 2019.

[169] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.

[170] A. Kendon, *Conducting Interaction: Patterns of Behavior in Focused Encounters*. Cambridge, U.K.: Cambridge University Press, 1990.

[171] C. Gloor, "Pedsim: A microscopic pedestrian crowd simulation system," 2003.

[172] 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.

[173] J. S. Coleman and J. James, "The equilibrium size distribution of freely-forming groups," *Sociometry*, vol. 24, no. 1, pp. 36–45, 1961.

[174] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv preprint arXiv:1707.06347.

Vita

Kapil Katyal is completing his Ph.D. in Computer Science at Johns Hopkins University where he works on machine learning and prediction to improve robot navigation. He received his B.S. degree in Computer Science and Engineering from Pennsylvania State University where he was a member of the Schreyer Honors College and an M.S. degree in Electrical Engineering from Columbia University. He is currently a robotics researcher and supervisor of the Robotic Intelligence section at the Johns Hopkins University Applied Physics Lab (JHU/APL). At JHU/APL, he leads several robotics and AI projects related to robot navigation, terrain traversability, pedestrian and group detection and continual learning algorithms. Upon graduation, Kapil will continue at JHU/APL as a principal robotics researcher and section supervisor.