Interpretable Monitoring for Self and Socially Aware Mobile Robot Planning

---

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Rahul Peddi

December 2022

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Rahul Peddi

This Dissertation has been read and approved by the examing committee:

Advisor: Nicola Bezzo

Advisor:

Committee Member: Tariq Iqbal

Committee Member: Madhur Behl

Committee Member: Arsalan Heydarian

Committee Member: Tomonari Furukawa

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

December 2022

Interpretable Monitoring for Self and Socially Aware Mobile Robot Planning

by

Rahul Peddi

B.S., Mechanical and Nuclear Engineering, Virginia Commonwealth University, 2017

Abstract

Autonomous mobile robots (AMR) are rapidly being introduced into our world in transportation, delivery, medical service, agriculture, and household applications. Their ability to reduce the burden on humans has made them viable and increasingly popular sources of productivity, but with their increased presence in our society, assuring that they behave in socially acceptable and safe ways is critical to their widespread integration and success. However, in many real-world applications, these robotic systems are subject to various uncertainties from different sources, such as the presence of dynamic actors like humans and other robots, or the presence of external disturbances and sensing/actuation faults. These uncertainties bring challenges to motion planning as they can cause the robot to behave in strange and unnatural ways, deviating away from their desired behaviors, towards humans, and potentially into unsafe situations. Many of these uncertainties appear during robot operations, and robots typically use reactive motion planners, which may not be agile enough to keep the system or its environment safe. More recently, robot planning has been achieved through learning-based methods, which may have proactive components, but these approaches use black boxes, making it challenging to understand why and how robots make certain decisions, which is critical to gain trust and fully integrate robots into our shared world.

This dissertation presents a set of proactive motion planning frameworks that promote social awareness and safety for autonomous mobile robots operating under uncertainties. The frameworks we develop monitor the future states of mobile robots and the nature of future interactions between robots and dynamic actors to improve and refine motion planning accordingly to a given scenario; whether it is a social navigation case study in which the robot must safely navigate in the presence of multiple humans, or if uncertainties are coming from different sources like sensing/actuation faults.

First, we introduce a Hidden Markov Model (HMM)-based predictive model that adds to traditional prediction methods by accounting for uncertainties in predictions to plan proactive motion for a robot in the presence of multiple humans. Explicit predictions, however, can be restrictive in environments with multiple actors. To address this challenge, we reduce the problem to a binary classification through the design of a Decision Tree (DT)-based interpretable monitor that is used to predict and explain future interactions with dynamic actors for proactive non-interfering motion planning. We further extend this monitoring approach to adapt and improve on the failure modes of a baseline reactive motion planner, while also re-integrating HMM principles to enable fast runtime updating, so that predictions and planning can improve as operations continue. To extend our work to handle highly dynamic and dense environments, we leverage the idea of attention within human-human interactions and design a deep neural network (DNN) based approach that predicts which actors are most important to consider as constraints within the framework of a model predictive controller (MPC). Finally, we show that the approaches presented in our work can extend beyond the social navigation case study, through work in recovery from failures for a robot under decision uncertainties. The techniques presented in this dissertation are validated through extensive simulations and experiment case studies with real unmanned ground and aerial vehicles navigating in the presence of humans.

"Everyone you will ever meet knows something you don't"
– Bill Nye (The Science Guy)

Acknowledgements

First, I would like to express my deepest gratitude to my advisor, Nicola Bezzo, for his mentorship and support throughout my time at the University of Virginia. His endless guidance, support, and encouraging words have helped me become the researcher and thinker that I am today. His dedication to his students, not only as an advisor but also as a teacher is unmatched, and I'm thankful that I had the opportunity to teach with him and learn how to teach from him throughout my graduate studies. I will always be thankful for the five years that I have spent under his guidance.

I would also like to thank my dissertation committee members: Prof. Tariq Iqbal, Prof. Madhur Behl, Prof. Arsalan Heydarian, and Prof. Tomonari Furukawa for insightful and constructive feedback about my research as well as their support during my Ph.D. studies.

I would also like to express my gratitude to my friends and colleagues at UVA – Esen, Tony, Paul, Shijie, Carmelo, Jacob, Phil, Lauren, Nick, Will, Patrick, Pravardhan, Garrett, and Beatrice. I will forever cherish the conversations, laughs, and wonderful moments we've shared. Beyond work, I would like to thank my closest friends: Shashi, Kern, Vaibhav, Hannah, Ayush, Parth, and Simha (in no particular order, I promise) for all their support, love, and tolerance over the past few years – and also for the endlessly distracting, but much-needed group chats. Also, special thanks to those who did the NYT crossword with me every night for 532 straight days during the pandemic.

I would also like to thank my parents, Srinivas and Anitha, who deserve all the credit for who I've become today; they helped me develop a love for learning, always encouraged me to ask hard questions, and do my best at anything I do. I thank them and my sister, Sreya, for their persistent understanding, help, and love over the years. They have always been there for me, whether it's sending food and snacks, providing support during my lowest moments, or sharing my joy during my best moments. I am also grateful to my community and family friends in Richmond – their words of encouragement, well wishes, and advice (both solicited and unsolicited) go a long way. I

would also like to thank my family in India and in particular, my late grandfather, who is sorely missed. He always made it a point to call me and provide the best advice whenever I had a new publication or presentation.

Last but certainly not least, I'd like to thank my partner Talie for always being by my side, and providing unconditional love and constant support. She is an endless supply of positive energy and always knows how to turn a bad day around (usually through food). She is my best friend and the joy of my life, and I thank her for always assuring me that doing my best would be enough, even when I didn't believe it.

Funding: I acknowledge the National Science Foundation (NSF) through grants #1816591 and #1823325, and the Defense Advanced Research Projects Agency (DARPA) under Contract FA8750-18-C-0090.

Contents

List of Figures

List of Tables

List of Abbreviations

AMR	Autonomous Mobile Robot
DT	Decision Tree
DNN	Deep Neural Network
HMM	Hidden Markov Model
MLP	Multi-Layer Perceptron
MPC	Model Predictive Control(er)
RA	Reachability Analysis
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

Chapter 1

Introduction

Autonomous mobile robots (AMR), such as unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV) have become increasingly common in our daily lives: we see autonomous vehicles and delivery robots around us in major cities [49], we find large service robots performing cleaning or monitoring tasks in our hospitals and transit stations [73], and even households have started to accept robots as an integral part of the home through small appliance robots, such as robotic vacuum cleaners [98]. This is all possible due to technological advancements in robot design, sensing and perception, and computation, and in fact, using these autonomous robots in many scenarios has become advantageous over employing human workers or other technologies. Figure 1.1 depicts some examples of real-world applications of AMRs.

Figure 1.1: Autonomous mobile robotic systems are used for a variety of operations including package delivery, transportation systems, commercial applications, and agricultural purposes.

Overall, we as a society have begun to rely on these robots to perform tasks that are mundane or

dangerous for humans, such as warehouse operations [14], search and rescue operations in dangerous environments [72], or automated navigation and surveillance [24]. However, in many of these real-world tasks, robots interact with highly dynamic and uncertain physical environments. Assessing and overcoming these uncertainties becomes integral for mobile robots to successfully serve their purpose in the real world.

In this dissertation, the work is primarily focused on how AMRs can generate self and socially aware motion planning in such highly dynamic environments. We define socially aware motion planning as robot movement that respects common social norms, such as maintaining a comfortable distance between actors (human and robot, in our case) and inducing minimal deviations in the paths of nearby humans. To be successful in such settings, these AMRs must also be self-aware and proactively plan how they should interact with the humans in the environment. A challenge, however, is that these interactions carry a high degree of uncertainty, where unexpected human behaviors might ultimately lead to undesired robot motion and even failures, jeopardizing the robot's ability to complete its task. Specifically, robots must handle uncertainties about human intentions, future interactions the robot may have with nearby humans, and how behaviors of such humans change over time and in the presence of others. For example, in the motivational Figure 1.2, a robot navigating in the presence of a person can behave in a reactive way, through which it successfully avoids collisions, but exhibits strange and unnatural behavior that forces the person to alter their path, potentially sending robot and human into undesirable or even unsafe states.

On the other hand, people interact and cooperate with one another seamlessly, sharing the burden of avoiding and accommodating one another. This is because humans not only implicitly understand others' intentions and how we will interact with them in the future, but we also implicitly communicate our intentions by changing our behavior proactively, often well in advance of a possible collision. If a robot could similarly predict the intentions and future interactions of nearby actors and plan motion proactively, then it can behave in a more socially acceptable manner, making its integration into real-world scenarios smoother.

Towards predicting these uncertain interactions between a robot and dynamic actors and planning robot behaviors, end-to-end learning-based approaches have been shown to be powerful tools [64, 67, 30]. However, these approaches tend to use black-box models throughout the prediction and planning phases, making it difficult to understand why the robot decides to move in a certain

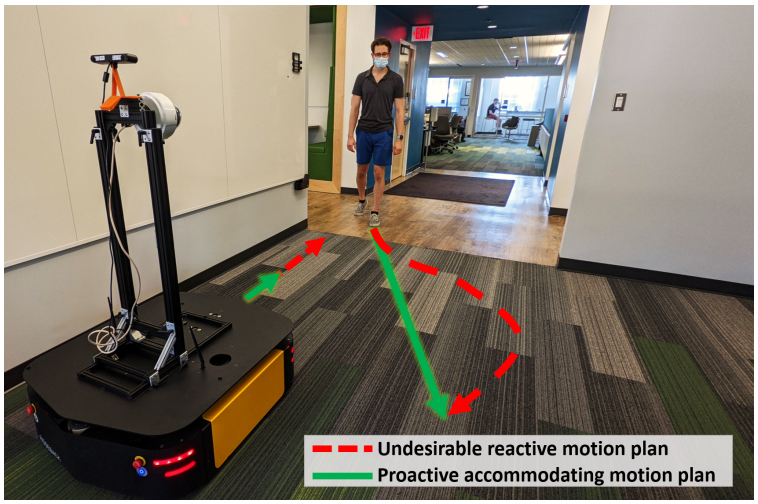


Figure 1.2: A reactive motion planner fails to account for the interaction between human and robot, causing the human to deviate from the desired path. By reasoning about the future interaction with the human, the robot is able to plan proactive and accommodating motion.

way. Under uncertainties, this lack of reasoning about robot prediction and planning can lead to unexpected behaviors or degraded performance, and it can compromise the safety of the robot and its surroundings. Let us take, for example, the Uber Autonomous Vehicle crash in 2018 [82]. The vehicle was not certain about the classification of a pedestrian, but made a decision anyway, which proved to be fatal. If the car were able to at least reason about the uncertainty in its classification, and include that reasoning in the decision-making process in some manner, the outcome may have been different. This type of outcome makes it evident that it is critical to design prediction and planning frameworks that can reason about predictions and associated uncertainties to proactively prevent unsafe situations, promoting safe robot motion under uncertainties.

Given these challenges, we present a blend of data-driven and model-based methods to efficiently and effectively handle uncertainties that mobile robots experience in the real world, while enabling the robots themselves and users of these AMRs to reason about the outcomes of predictions and planning behaviors for the robot. However, designing such approaches in a way that generalizes well to many scenarios is challenging as uncertainties can be caused by a number of different factors, such as the unknown behaviors of moving actors or the presence of sensing and actuation faults that compromise the robot's ability to complete its task.

In this dissertation, we provide several frameworks that consider this problem of a robot navigating in the presence of multiple dynamic actors and reasoning about uncertainties that are

attributed to the actors. In addition, there may be uncertainties in decision-making that need to be taken into account to guarantee that a system can safely operate around humans and in safety-critical operations. To this end, we show that the methods presented for social navigation via explainable monitoring can be extended to reason about uncertainties and demonstrate a case study on assessing prediction uncertainties about external disturbances or sensor and actuator faults that might be affecting a mobile robot.

With these considerations in mind, the objectives of this work are to solve the following challenges:

- How to predict future intentions of surrounding actors or future states of the system at runtime to inform proactive and safe uncertainty-aware motion planning.
- How to provide explanations for predictions and directly leverage explanations and reasoning to plan non-interfering, socially aware motion.
- How to enable robots to learn and model new behaviors of moving actors at runtime in highly dynamic environments in the presence of multiple moving actors.
- How to adjust predictions and robot behaviors to improve proactive motion planning in the presence of dense crowds of dynamic actors.
- How to include and account for uncertainties in robot decision-making to provide safe motion planning in the presence of failures caused by disturbances and faults.

In the rest of this chapter, we review the related work and the state-of-the-art in safe motion planning in the presence of dynamic actors and under uncertainties. We also overview our approaches to solve these problems, and lastly, we summarize the contributions of this dissertation.

1.1 Related Work

In this section, we provide an overview of related literature in prediction and motion planning in the presence of uncertainties. First, we start with a study of traditional motion planning for collision avoidance techniques, followed by learning-enabled prediction techniques, including interpretable machine learning for explainable predictions. We then discuss the literature on leveraging predictions for motion planning algorithms to generate desirable behaviors in the presence of dynamic actors.

We also examine the literature on extending such approaches for handling not only dynamic actors, but dense and highly dynamic crowds. We follow this with an overview of traditional and learning-enabled techniques used to recover failing robots with decision uncertainties, to connect prediction and planning to techniques to the general scope of robot motion planning under uncertainties.

1.1.1 Traditional Motion Planning for Collision Avoidance

The problem of motion planning for collision avoidance has been heavily studied in the literature of mobile robotics to enable safe autonomous navigation in the presence of static and dynamic obstacles. Traditional collision avoidance techniques include sampling-based and graph search approaches. The A* search algorithm [21] for graph search has been shown to provide optimal collision-free paths, but typically relies on a known environment where the positions of obstacles are known. In some approaches derived from A* [28], including D* [104] and D* Lite [95], unknown obstacles can be handled, but avoiding dynamic obstacles remains a challenge for graph search-based approaches. Sampling-based approaches such as the rapidly-exploring random tree (RRT) [59], which constructs a random tree to find a collision-free path, are used for a broad range of path planning problems. Extensions of the RRT approach such as RRT* [74] and RRT*-smart [48] have been leveraged for mobile robot navigation tasks, however, these approaches tend to scale poorly and can have computational issues when handling multiple dynamic obstacles. The dynamic window approach (DWA) [36] is another widely-used sampling-based algorithm, in which the search space is restricted by dynamic feasibility constraints and the position of obstacles, making it viable for avoiding dynamic obstacles at runtime. We compare some of the work presented in this dissertation with DWA to further assess its viability in the social navigation case study.

In addition to sampling and graph-based approaches, physics-based approaches, such as velocity obstacles [34] and artificial potential fields (APF) [53] have also been designed to avoid dynamic obstacles in the environment. Velocity obstacles and their advancements, reciprocal velocity obstacles [99, 54] and optimal reciprocal collision avoidance [100], have shown to perform best when multiple agents are being controlled in the same manner, but do not account for uncertainties and scale poorly to handling different behaviors of dynamic obstacles [5]. Artificial potential fields are more flexible without making assumptions about the behaviors of dynamic obstacles, and have been leveraged for mobile robot navigation in a number of applications [68, 40, 86]. These approaches are

very efficient and scale well to handling to multiple obstacles, but do suffer from the local minima problem, particularly when dynamic obstacles are present in the environment, and a number of approaches have been proposed to escape said local minima [20, 75]. While such approaches are efficient and effective, a missing component is the ability to predict the future states of dynamic obstacles, and leverage these predictions for motion planning.

1.1.2 Prediction and Motion Planning

Most traditional reactive planning methods are not sufficient without some proactive component that performs predictions about the future states of dynamic obstacles and the robot itself. Several approaches using graph search and directional rule-based methods [83, 44] have been used to provide safe motion planning given strict a priori assumptions of actor trajectories based on heading and velocity. These techniques typically predict or expect dynamic actors to behave in a certain way and typically do not account for uncertainty and are unable to learn or update the expectations at runtime, often resulting in "freezing" [97, 90] behaviors when dynamic actors behave differently from what was expected. Reachability-based and confidence-based approaches [11, 35] rely on simplified dynamic models to predict human motion to plan safe and proactive motion around dynamic actors, while updating at runtime its belief about the expected goal of the actors. Reachability analysis performs well in its representation of uncertainty surrounding dynamic actor motion and control policy computation, however, it suffers from computational scalability [61], which affects its applicability in cases with many actors.

Recent developments in machine learning algorithms such as deep neural networks and deep reinforcement learning have enabled these techniques to be used for avoiding collisions with dynamic obstacles. The work in learning-enabled prediction involves using neural network architectures to predict the future positions/paths of dynamic obstacles [47, 67]. Methods such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs) [47, 67] have made substantial progress in this area by predicting the motion of nearby actors, and leveraging these predictions to generate safe robot motion. There are also a number of Deep Reinforcement Learning based approaches [31, 19, 56] used to attain socially acceptable behaviors, and in [106], the authors use deep neural networks (DNNs) to achieve similar results. While these methods are effective for predicting actor trajectories and generating good robot behaviors, they contain complex network architecture

and as a result, require a dedicated training phase to improve robot behaviors. Moreover, it is difficult to understand the mapping from input states to prediction, and thus, it is difficult to reason why a resulting behavior is appropriate or correct for the robot's task.

This extensive use of ML techniques brought about the need for explaining how these black-box models work [42]. The existing approaches were split into three categories: model explanation, outcome explanation, and model inspection, each of which aims at explaining the model itself, the reason for an output given the input, and the sensitivity of the model with respect to changing the input, respectively. These approaches, however, provide explanations that are hard to understand from a logical point of view and do not provide counterfactuals. To logically explain the reasoning behind the outputs of ML models, authors in [87, 41] provide techniques to find local reasoning as to why a data point was assigned to a certain class in simple classification learners, such as decision trees [16] (DT) with decision rules and reasoning that are easier to understand for humans.

1.1.3 Socially Aware Navigation in Dense Crowds

With the integration of mobile robots into our world, there has been widespread interest and research in enabling robots to navigate in dense crowds in a socially acceptable manner. Some of the works discussed in the previous sections [31, 19] begin to address this problem, but have a rather small "maximum" number of actors (3-5) they can consider at runtime. Typically, these types of works use a distance heuristic and consider some number of actors closest to the robot to inform planning [69]. While this may be viable, other research has suggested crowds are highly dynamic and tend to agree with the notion of limiting the focus of the robot [89, 90, 18, 91, 101], but each approach it from a different perspective. Presented in [89] is a deep reinforcement learning (DRL) method to imitate human behaviors in large crowds, but considers an exhaustive reachable set for human behaviors, which may be impractical and unnecessary. Similarly, [90] presents the notion of a "freezing zone" that the robot must avoid, and planning is also achieved through DRL. Authors in [91, 18] focus directly on limiting the actors through grouping and attention mechanisms. These approaches, however, use end-to-end learning and black boxes to move directly from perception to planning, which makes it difficult to understand why the robot is behaving in a certain manner. However, attention remains an interesting concept, as it provides an avenue for limiting the challenges of navigating in very large crowds. In [101], authors present a model predictive controller (MPC)

that takes into account group behavior and learns to consider all actors and model them explicitly into groups, around which a provably safe MPC planner can be used. We focus on this idea in this dissertation, but blend the notion of attention with a strong baseline MPC controller that takes into account the dynamics of the actors. Furthermore, by including the MPC, instead of using DRL for robot control, we are able to reason about the planning outcomes based on the dynamics of the robot and actors.

1.1.4 Decision-making Under Uncertainties

Decision-making for AMR has become a well-studied problem over the years [107], but safe decision-making under uncertainties remains an open challenge. Many recent approaches use learning-enabled components, such as deep neural networks (DNN) [58] and deep reinforcement learning (DRL) [56], to make quick decisions with a reasonable level of accuracy for many applications [92]. However, a vast majority of these techniques do not consider uncertainties and return only one decision, which might be incorrect in the presence of measurement or process noise at runtime [84]. Significant effort has been devoted to achieving uncertainty-aware decision-making with machine learning; authors in [51, 2] use sampling-based methods, such as bootstrapping or Monte Carlo sampling with DNNs. However, the effectiveness of sampling-based methods for uncertainty evaluation depends heavily on the quality and number of samples taken and can become too computationally expensive for robot control in many cases [65]. Moreover, methods using DNN and DRL contain black boxes, which make it difficult for a user to understand why a particular decision was made, which has been shown to improve the overall performance of decision-making systems [88]. Other approaches make considerations on the training dataset through variational inference [94] on the training data and active learning [96] to perturb and gain more information about decisions. We take inspiration from these approaches and integrate them into our work in failure detection and recovery under uncertainty.

As for detecting and recovering from sensor and actuator failures [1], control theorists have proposed a number of approaches, in which detection often relies on state estimation [43] and deviation/bias measurement and analysis [52], which are easy to understand and work well for detecting different degrees of a particular type of failure, but do not extend well to detecting different failures that can appear similar, thus making learning-based approaches more appealing [102].

The control techniques used for correction include adaptive control [50] and model predictive control (MPC), which has been shown extensively to produce safe motion planning under degraded conditions [108]. However, the black-box issue remains, and this brings about the desire to bridge the gap between learning and control-based approaches to design a method to keep the system safe under different sensor and actuator faults and disturbances.

1.2 Overview of the Research

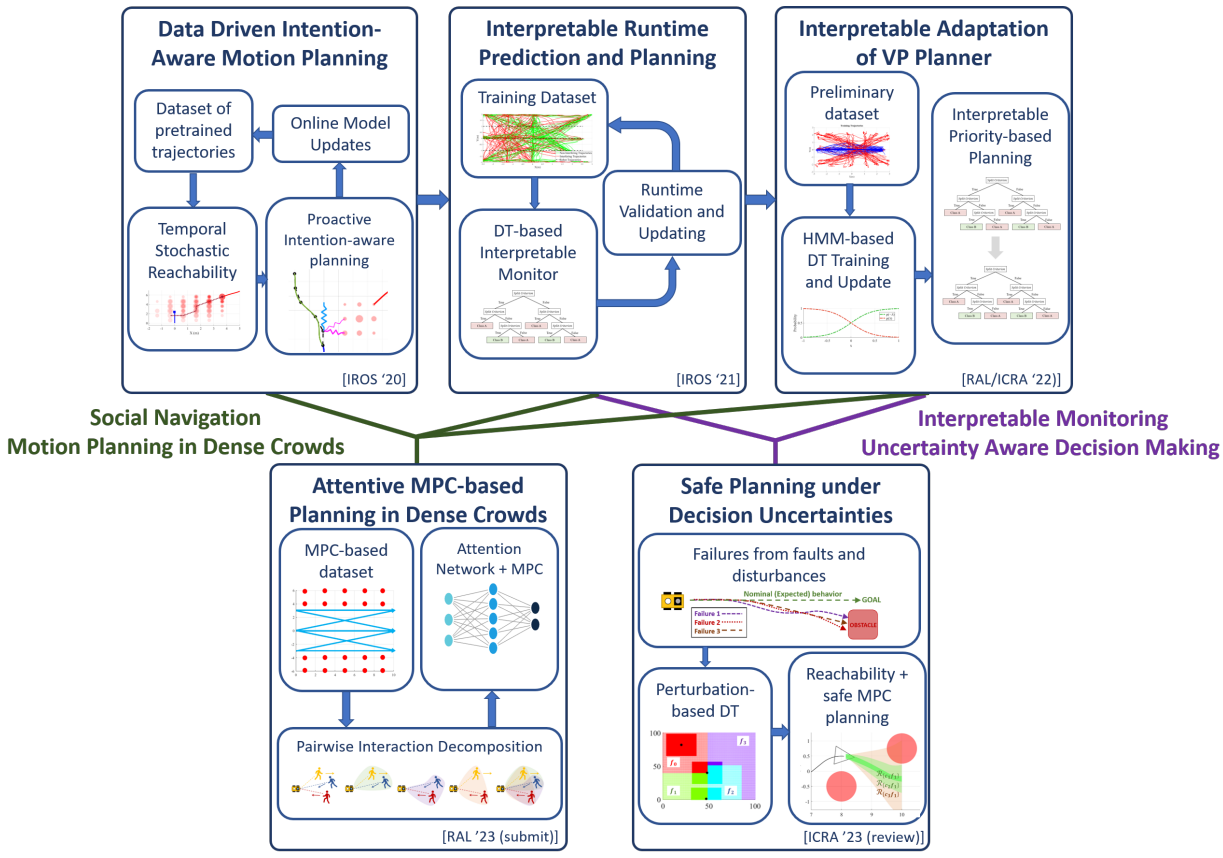


Figure 1.3: Overview of the research presented in this dissertation.

The research presented in this dissertation consists of five main segments that include: 1) explicit predictions of dynamic actor paths for proactive robot planning, 2) interpretable monitor design for prediction and planning in co-robotic environments, 3) interpretable and proactive adaptation of a reactive VP planner at runtime to improve social navigation, 4) attention-aware social navigation in dense crowds, and 5) extension of interpretable uncertainty aware techniques to multi-class scenarios

with an application in planning under disturbances/faults. Figure 1.3 provides an overview of the research presented in this dissertation, which is described in detail in the next section.

1.3 Dissertation Organization and Contributions

In this section, we present the composition of this dissertation by providing summaries of each chapter and specifying contributions within each chapter. Chapter 2 focuses on explicitly predicting future states of dynamic actors to plan proactive robot motion, Chapter 3 introduces the design of an interpretable monitoring technique that bypasses explicit predictions for motion planning in co-robotic environments, Chapter 4 further describes how interpretable monitoring can be used to adapt and solve deficiencies of well-known and widely used motion planners for social navigation, Chapter 5 focuses on expanding prediction and planning approaches to dense, dynamic crowds, and Chapter 6 demonstrates how the techniques developed and utilized throughout this dissertation can be extended to other domains in which uncertainties have an impact on robot motion planning.

Chapter 2: Data-driven Proactive Intention-Aware Social Planning

In this chapter, we discuss our novel data-driven framework for proactive intention-aware robot motion in presence of multiple moving humans. We make explicit predictions about the future positions of humans and assess the uncertainties around the predictions to provide context for robot motion planning. Unique from other approaches, our uncertainty assessment is leveraged to create stochastic reachable sets which are paired with a temporal virtual physics-based planner for proactive planning. We also provide a technique for updating the predictive model at runtime by collecting and incorporating observations of new and unexpected behaviors. We validate our framework with simulations and experiments consisting of a UGV navigating proactively through environments containing multiple humans. This chapter is based on the publication:

- R. Peddi, S. Gao, C. DiFranco, and N. Bezzo, "Data-driven Framework for Proactive Intention-Aware Motion Planning of a Robot in a Human Environment," *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020.*

Chapter 3: Interpretable Runtime Prediction and Planning in Co-Robotic Environments

In this chapter, we relax the requirement of explicitly predicting the future paths of dynamic actors. We introduce the design of an interpretable monitoring approach for robot decision-making that reduces the social planning problem to a binary classification about whether the robot will interfere with the future paths of nearby dynamic actors. Different from prior research, our interpretable monitor is used to provide explanations and counterfactuals for the interfering predictions, providing reasoning for the prediction and providing options for how the robot can correct its interfering behaviors. The prediction, explanation, and counterfactuals are used directly to provide a high-level robot motion plan, which is then sent to a pure pursuit controller for low-level controls. Also unique from other research, our framework is able to collect observations at runtime to update and improve the predictive model and refine the behaviors of the robot. Our framework is validated with multiple simulations and experiments consisting of a UGV navigating through environments containing multiple humans without interfering with their paths. This chapter is based on the publication:

- R. Peddi and N. Bezzo, "Interpretable Runtime Prediction and Planning in Co-Robotic Environments," *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2021.*

Chapter 4: Interpretable Adaptation of Motion Planners for Social Navigation

This chapter expands our interpretable monitoring approach to correcting the failures of well-known and widely used virtual physics-based planners, such as artificial potential fields. We note that such planners are efficient and scale well to avoiding collisions with multiple agents, but suffer from local minima issues, which in dynamic environments can cause interference with nearby actors. We leverage the interpretable monitor introduced in Chapter 3 to adapt the parameters of such planners to ensure non-interfering behaviors. We additionally design a new method for the robot to update and improve predictions and planning faster than the data collection methods in the previous chapters. Furthermore, we extend the efficacy of the counterfactuals to adapt corrective actions to the robot's priorities, as we account for different desired behaviors for robots operating in different environments with different applications. Our framework is validated with multiple simulations and experiments consisting of a UGV and a UAV equipped with the virtual physics

planner that successfully navigates without interfering with the paths of multiple dynamic actors. This chapter is based on the publication:

- R. Peddi and N. Bezzo, "An Interpretable Decision Tree-based Virtual Physics Method for Non-interfering Social Planning," *in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2022 and in the IEEE Robotics and Automation Letters (RA-L).*

Chapter 5: Attention-aware Robot Social Planning

In this chapter, we expand on previous approaches by considering interactions between actors in order to achieve socially aware robot motion planning in dense crowds. We demonstrate that by modeling human-human interactions in dynamic crowds, the robot may be able to consider only a subset of the actors, without the need to include every actor, in order to navigate successfully through the crowd. We introduce an MPC framework that can predict the future states of the robot and nearby actors based on the respective dynamic models, but we note that considering all actors in a very dense environment can be too computationally expensive or intractable. To solve this issue we design a predictive model that directly takes into account the states of nearby actors to identify the minimal set of actors the robot should pay attention to. We show that this framework generalizes and is successful for large, dense crowds in small spaces through extensive simulations of a UGV navigating through crowds of different topologies. This chapter is based on the paper:

- R. Peddi and N. Bezzo, "Attentive Model Predictive Control for Crowd-aware Robot Navigation," *IEEE Robotics and Automation Letters (RA-L) (in preparation)*

Chapter 6: Interpretable Monitoring and Recovery Under Decision Uncertainties

In this chapter, we consider that predictions from learning-enabled components can occasionally be incorrect due to noise and other uncertainties, and that considering only a single prediction may be unsafe. We extend the previously proposed monitoring approaches to deal with this issue by directly considering uncertainties in predictions to inform safe motion planning. Furthermore, we demonstrate that the proposed explainable monitoring methods can be extended beyond social planning applications: mobile robots can often operate in uncertain environments and have a number of sensors and actuators for perception and control, all of which can be the cause of failures; either

through external disturbances or sensing/actuation faults. We train our interpretable monitor for detecting failures on an offline training phase with different failures and predefined controllers that take into account different dynamics under each failure. At runtime, we not only detect and explain which failure may be affecting the system, but also introduce a local perturbation-based approach that considers prediction uncertainties to identify the safest controller to use even when the monitor is uncertain about the cause of the failure. This approach is validated with both simulations and experiments of a UGV experiencing different failures at runtime. This chapter is based on the publication:

- R. Peddi and N. Bezzo, "A Decision Tree-based Monitoring and Recovery Framework for Autonomous Robots with Decision Uncertainties," *IEEE International Conference on Robotics and Automation (ICRA) 2023 (submitted)*

Chapter 7: Conclusions and Future Directions

In this chapter, we conclude the dissertation by summarizing the results from all the aforementioned works and discussing potential future directions to build on.

1.4 Summary of Contributions

To summarize, the work presented in this dissertation will contribute to the existing state-of-the-art in robot motion planning under uncertainties and in the presence of dynamic actors by providing:

- A data-driven Hidden Markov Model-based approach to predict the future states of dynamic actors to compute temporal stochastic reachable sets, which are used to generate proactive and accommodating motion planning.

- A novel Decision Tree-based framework to make interpretable predictions about future interfering interactions between a robot and surrounding dynamic actors, bypassing the need to make explicit predictions at runtime. The predictions, explanations, and counterfactuals are directly used for robot planning and control.

- An interpretable prediction and explanation method with fast runtime model updating to augment and compensate for the local minima problems in Virtual Physics based planners with application in non-interfering, priority-aware navigation in the presence of dynamic actors.

- A deep learning-based approach for attentive, non-interfering crowd navigation with a model predictive controller that takes into account human-human interactions and bypasses the need to consider as a constraint every single actor in dense crowds.

- An uncertainty-aware interpretable prediction approach paired with model predictive control to deal with the decision uncertainties about external disturbances, noises, or sensing and actuation faults that can cause the system to fail.

- A final contribution of this dissertation is in the extensive implementations with realistic simulations and real-world experiments with UGVs and UAVs. In conjunction with the experiments on real hardware, this thesis also provides training datasets containing a human navigating in the presence of a mobile robot [77].

Chapter 2

Data-driven Proactive Intention-Aware Social Planning

In this chapter, we introduce our data-driven framework for proactive intention-aware motion planning for autonomous mobile robots in environments. This framework leverages Hidden Markov Model (HMM) theory to predict the future states of dynamic actors and a control scheme based on temporal virtual physics to plan safe and proactive robot motion. This approach is further supplemented with stochastic reachability analysis to identify multiple possibilities of actor future states and a method to concurrently learn, update, and improve the predictive model with new observations at runtime. We validate this approach with both simulations and experiments focusing on a UGV goal navigation problem in environments consisting of multiple humans. This work has been published at the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

2.1 Introduction

In many cases, autonomous mobile robots that share space with humans perform their missions by treating the surrounding humans as stationary obstacles. People, in turn, are expected to work around the robot, and are required to learn to adapt to the robot that is often moving in unnatural ways. Humans have to change their behaviors in response to what the robot is tasked to do. People, on the other hand, interact and cooperate with one another nearly seamlessly. This is because

humans not only implicitly understand others' intentions, they also implicitly communicate their intentions by changing their behavior proactively, often well in advance of a possible collision.

In this chapter, we address the problem of enabling proactive intention-aware motion on a robot that coexists with humans in environments like airports, train stations, labs, and offices. In our case, *proactive intention-aware motion* refers to robot motion that accommodates the human's future motion while maintaining the robot's desired goal, which brings about the following challenges:

1. How to efficiently and correctly predict the intentions and uncertainties about the intentions of surrounding dynamic actors.

2. How to use uncertainty-aware intention predictions to plan motion that proactively avoids collisions with the dynamic actors.

To predict the intentions of surrounding actors, we use a data-driven Hidden Markov Model-based technique along with temporal virtual physics (VP) based planning based on a spring-mass-damper system to generate proactive, accommodating robot motion. Furthermore, the prediction model is updated at runtime by collecting observations to further improve robot behaviors as operation continues. In Fig. 2.1 we show a pictorial example, in which the robot predicts forward the future states of the actors to determine how it should move through the environment.

Figure 2.1: Pictorial representation of the motivation behind this work. Our approach computes temporal stochastic reachable sets and plans motion that proactively accommodates human intentions.

2.2 Problem Formulation

Consider a mobile robot tasked to go from an initial position $\boldsymbol{q}_0$ to a goal $\boldsymbol{q}_g$ while negotiating and accommodating its motion with surrounding actors, in particular humans, $h_i \in S_h(t)$, $i = 1, \ldots, n_h$, where $S_h(t)$ is a time varying set of n_h humans in sensing range with the robot. Each human in the environment is following some trajectory from an initial position $\boldsymbol{q}_i^{start}$ to an a priori unknown (to the robot) goal $\boldsymbol{q}_i^{goal}$. We assume that the robot can localize itself within the environment and distinguish between static obstacles and humans using on board and external sensors with standard localization and sensing techniques [7]. The dynamics of the robot can be represented in the typical state space form, $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{x} \in \mathbb{R}^n$ is the state and $\boldsymbol{u} \in \mathbb{R}^m$ is the control input. With such premises, the robot has the objective to predict the intended motion of other actors in its sensing range and proactively plan its motion to minimize the human changes in path due to the presence of the robot. In doing that the robot is also implicitly communicating with the humans to acknowledge that it understand their intention. Formally, the problem is:

Problem 2.1 (Intention Aware Proactive Motion Planning and Control) *Consider a robot navigating in an environment with other humans. Given a set of observed humans S_h, the objective is to predict their reachable states $\mathcal{R}_h(t)$ at runtime over an horizon H, find a policy to anticipate future robot reactions, and plan a trajectory to minimize human deviating maneuvers created by the robot, such that:*

$$||\boldsymbol{q}(t) - \boldsymbol{q}_i(t)|| > \Delta, \forall i \in S_h(t), t \geq 0 \tag{2.1}$$

where $\boldsymbol{q}(t)$ and $\boldsymbol{q}_i(t)$ are the position of the robot and the i^{th} human at time t respectively, and Δ is minimum safe distance.

A secondary problem that we propose to investigate in this work is how to improve predictions over time. To this end we use the theory of Hidden Markov Model (HMM) to collect a history of observations and emission probabilities that are constantly updated online to consider runtime behavior that either was never observed during training or to reinforce/change the expected future predictions.

2.3 Prediction and Planning Framework

In this section we describe the framework adopted for prediction and control of a robot in a human environment. Specifically, we follow the architecture depicted in the block diagram in Fig. 2.2.

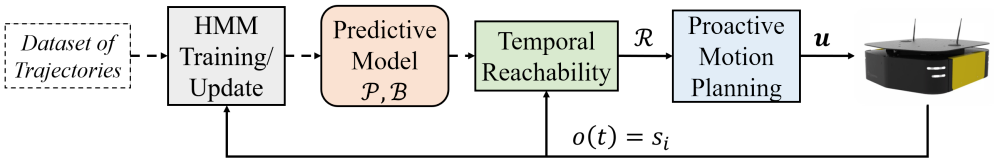


Figure 2.2: Block diagram of the HMM-based data-driven intention-aware motion planning

At the core of our framework we consider a HMM that is trained using offline data and constantly updated at runtime for more accurate predictions and to include new behaviors. The predictive model generated by HMM training consists of a set of state observations $\mathcal{O}$ and emission matrix $\mathcal{B}$. Prediction is then executed by building a temporal reachable set $\mathcal{R}$ over a finite time horizon to include future human states and the associated probabilities.

Then, the motion planner that we propose here will consider this prediction to find reactions (deviations from the desired trajectory) of the robot at different time steps and proactively adjust the robot actions toward the point in the trajectory where a deviation was predicted. At runtime new observations are added to $\mathcal{O}$ and are then used to update the HMM, improving and refining $\mathcal{B}$, and thus improving future predictions and inferences.

While we decide to utilize an initial offline training phase, the proposed framework can also work in the absence of offline training due to online updates. The inclusion of a library of trajectories and offline training results in more informed predictions and reachability analysis from the start, allowing us to better demonstrate the effectiveness of our motion planner. In the next sections, we describe in detail each component of our framework.

2.3.1 HMM-based Training

To train the predictive model for human intentions we propose a Hidden Markov Model (HMM) [103] which can be described by the tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{C}, \mathcal{G}, \mathcal{P}, \mathcal{B} \rangle$ where:

- $\mathcal{S} \in \mathbb{R}^N$ is the state space of the system which includes a finite set of unique states $s_i \in \mathcal{S}$, $i = 1, \ldots, N$ that can be visited by the system.

- $\mathcal{O} \in \mathbb{R}^T$ is a finite set of observations $o(t) \in \mathcal{O}$ collected over a finite past time horizon T, i.e., $\mathcal{O} = \{o(t-T), o(t-T+1), \ldots, o(t)\}$ and such that $o(t) = s_i \in \mathcal{S}$. Note that the states observed during T, $\mathcal{S}' \in \mathbb{R}^n$ are a subset of $\mathcal{S}$, $\mathcal{S}' \subseteq \mathcal{S}$ and $n \leq N$.

- $\mathcal{C} \in \mathbb{R}^T$ is the finite set of emissions, or inferences $c(t)$ that relate to the action taken each state, and $\mathcal{C} = \{c(t-T), c(t-T+1), \ldots, c(t)\}$.

- $\mathcal{G} \in \mathbb{R}^M$ is a finite set of M unique inferences that $\mathcal{C}$ can obtain, and $g_k \in \mathcal{G}$, where $k = 1, \ldots, M$, with $M \in \mathbb{N}$.

- $\mathcal{P} \in \mathbb{R}^{N \times N}$ is a transition probability matrix, that describes the probability of entering a certain state, $s_j \in \mathcal{S}'$, while currently in observed state $s_i \in \mathcal{S}'$, denoted as $\mathrm{P}(s_i \rightarrow s_j)$.

$$p_{ij} = P(s_i \rightarrow s_j) \tag{2.2}$$

Each transition probability p_{ij} is initialized as $1/N$. Transition probabilities are calculated by counting the occurrences of each state transition over all transitions from that state:

$$p_{ij} = n_{ij}/n_{i*} \tag{2.3}$$

where N_{ij} is the total number of transitions, $s_i \rightarrow s_j$, over T and N_{i*} is the total number of transitions from s_i to any state, and $N_{ij} \leq N_{i*} \leq T$. The state transition matrix is right-stochastic, meaning that the sum of all rows is 1 and is of the form:

$$\mathcal{P} = \begin{bmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \\ p_{N1} & & p_{NN} \end{bmatrix} \tag{2.4}$$

- $\mathcal{B} \in \mathbb{R}^{N \times M}$ is the emission matrix, which lists the probability b_{ik} of obtaining emission g_k given state s_i:

$$b_{ik} = P(g_k(t+1)|s_i(t)) \tag{2.5}$$

where $i = 1, \ldots, N$. Emission probabilities are initialized as $1/M$, and are calculated as follows:

$$b_{ik} = n_{g_{ik}} / n_{g_i *} \tag{2.6}$$

where $n_{g_{ik}} \leq n_{g_i *} \leq T$.

$$\mathcal{B} = \begin{bmatrix} b_{11} & \ldots & b_{1M} \\ \vdots & \ddots & \\ b_{N1} & & b_{NM} \end{bmatrix} \tag{2.7}$$

The general pictorial representation of an HMM is shown in Fig. 2.3. In this image, nodes labeled s represent observed states ($\mathcal{S}'$), while those labeled g represent the emissions ($\mathcal{G}$).

Differently from a traditional HMM, the "hidden" part applies to the emissions, rather than the states, which are directly observable (i.e., measurable) and therefore, have observable transitions. Another key difference is that we utilize the set of observations $\mathcal{O}$, to make informed predictions for stochastic reachability in addition to the emission matrix, which provides important behavioral information used for motion planning. Because of the human-robot interaction problem considered in this work, the specific states recorded in $\mathcal{O}$ are:

$$\boldsymbol{s} = \begin{bmatrix} d_x & d_y & \theta \end{bmatrix} \tag{2.8}$$

where d_x, d_y, and θ are the relative x-y positions and heading of a person in the robot local frame, respectively.

On the other hand, the emissions will capture qualitative inferences on how a human will behave when approaching the robot: specifically in this work we are interested in predicting whether a human starting from a state $\boldsymbol{s}_i$ will ***cross*** or ***not cross*** the robot's path at some point in the future during the operation, which impacts the way we plan robot motion; we want to accommodate more to a person who is crossing the robot's path.

The initial offline HMM training shown in Fig. 2.4 is executed on a set of trajectories specifically designed to capture the accommodating and avoiding behaviors of a humans moving from random initial positions to random goals in the environment around a robot that ignores the human and just follows a straight path starting from $(0, -2.5)$m to $(0, 3)$m at $v = 0.5$m/s. This training dataset

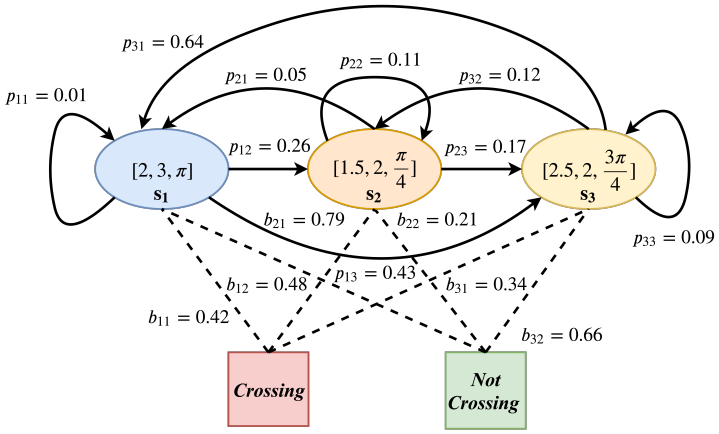


Figure 2.3: Diagram of 3 states within the Hidden Markov Model (HMM) used our framework

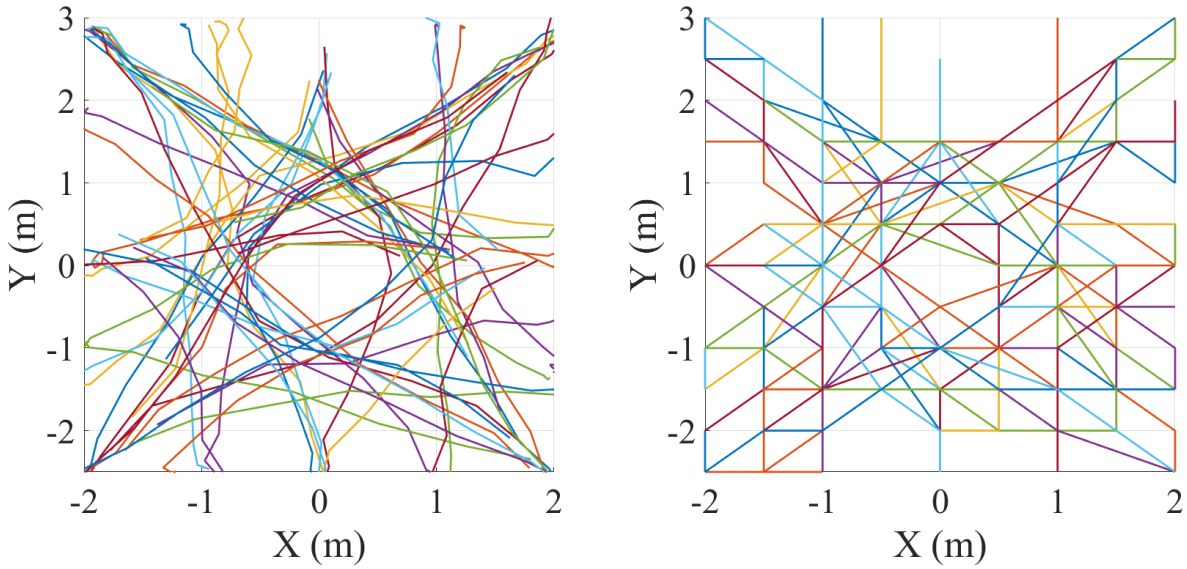
is included as "hmm_dataset.7z" in the GitHub repository in [77].

Collisions are not included in our training set because we assume that people in general do not behave adversarially, and therefore will not intentionally try to collide with the robot. The state vector (2.8) is discretized to prevent an infinite or exploding state-space. The dimension and discretization of the state space are selected based on the capabilities of the robot. A larger state space and finer discretization results in better approximation, but at an increased computational complexity. Note that increasing the state space too much also increases the uncertainties associated with the motion of the humans and hence it will generate overly conservative planning for the robot.

At runtime, observations are rounded to the nearest state in the discrete state space. Then, future state predictions and reachable sets are computed with the approximated state, and are geometrically transformed back to the observed state. Executing the training procedure creates a set of observations $\mathcal{O}$ and emission matrix $\mathcal{B}$, which serve as the predictive model for the likelihood of future states and implicit behaviors given a certain state observation at runtime.

2.3.2 Online Prediction and Stochastic Reachability

At runtime, given an observation of a person's state, $o(t) = s_i \in \mathcal{S}'$, we predict all possible future states and associated probabilities over a finite horizon H to enable proactive motion planning. A low H (e.g., $H = 1$s) can lead to predictions that are ineffective for proactive motion planning resulting in mostly reactive behaviors, while a very large H can be wasteful, as uncertainties can



(a) Training trajectories.

(b) Discretized training trajectories.

Figure 2.4: (a) Set of real trajectories used for the training process in the experiments. (b) Discretized trajectories according to a grid with cells 0.5m wide.

grow too large with time in continually evolving and unstructured environments.

To obtain the future states and probabilities over the selected H, we propose to use Reachability Analysis (RA), which is the process of computing the set of all reachable states for a system by taking into account its dynamic model and state transitions over a future horizon H [10]. The collection of all reachable states at a certain future time forms a *reachable set*, while a reachable tube is a temporal sequence of reachable sets. In addition to reachable states, a *stochastic reachable set* includes probabilities associated to each state [11], but generating such stochastic reachable sets can become computationally complex [61].

To perform such stochastic reachability analysis, we consider first the finite set of observations, $\mathcal{O}$. At runtime, we consider the future states and associated conditional probabilities of all the paths observed during training originating from the observed state at time t, $s_i(t)$ over an interval $[t+1, t+H]$. The probability of any reachable future state $s_j \in \mathcal{S}$ is given by $p(s_j(t+1)|s_i(t))$ and this is computed for any state along the path conditioned on the previous states. To limit the computation complexity of such approach for every state we maintain a list of the N_h most recent paths initiating from that state, obtaining a maximum of $N \times N_h$ trajectories that are used to perform such prediction. In this way, we also remove old and obsolete data and consider only the

most recent data in our prediction.

While $p(s_j|s_i)$ is readily available from the transition matrix $\mathcal{P}$, we cannot condition on the original state of the paths using only $\mathcal{P}$, due to the memoryless property of Markov processes. Instead, we leverage the set of observations, $\mathcal{O}$ to obtain the desired conditional probabilities.

We first identify the future states by searching for the N_h most recent instances of s_i in $\mathcal{O}$, followed by taking the subsequent H states in the set of observations. From this, we obtain n_{s_i} reachable paths of future states that originate from each instance of s_i. The collection of all reachable paths for a human in state s_i over the entire horizon forms the reachable tube, $\mathcal{R}^h_{s_i} \in \mathbb{R}^{n_{s_i} \times H}$:

$$\mathcal{R}^h_{s_i} = \begin{bmatrix} s_1(t+1) & \dots & s_1(t+H) \\ & \vdots \ddots & \\ s_{n_{s_i}}(t+1) & \dots & s_{n_{s_i}}(t+H) \end{bmatrix} \tag{2.9}$$

where each column of $\mathcal{R}^h_{s_i}$ consists of reachable states at the respective time-step.

Then, the probabilities of each temporal future state $s_j(t+\tau), \forall s_j(t+\tau) \in \mathcal{R}^h_{s_i}$ are computed by counting its occurrences as follows:

$$p(s_j(t+\tau)) = n_{s_j(t+\tau)}/n_{s_i} \tag{2.10}$$

where $n_{s_j(t+\tau)}$ is the number of paths that contain state s_j at time $t+\tau$, and $\tau \in [1, H]$.

We finally consider the emission matrix, $\mathcal{B}$, to infer the likelihood that each state $s_j \in \mathcal{R}^h_{s_i}$ in the reachable tube will lead to a crossing behavior: $p_e(s_j) = \mathcal{B}_{s_j,1}$.

This procedure is repeated over the horizon H, to obtain the most likely sequence of states and associated probabilities that are appended to $\mathcal{R}^h_{s_i}$. Reachable tubes for each human and associated probabilities are concatenated and stored to create one reachable tube, $\mathcal{R} \in \mathbb{R}^{n \times H}$, that encompasses all n possible paths for all sensed humans. Then, $\mathcal{R}$ is deconstructed into temporal reachable sets to be used for motion planning:

$$\mathcal{R}(t+\tau) = \begin{bmatrix} s_1(t+\tau-1) & s_1(t+\tau) \\ \vdots & \\ s_n(t+\tau-1) & s_n(t+\tau) \end{bmatrix} \tag{2.11}$$

where $\tau \in [1, H]$, and $\mathcal{R}(t+\tau) \in \mathbb{R}^{n\times 2}$. Note that a temporal reachable set $R(t+\tau)$ includes reachable states at $\tau - 1$ and at τ to capture the motion between two consecutive time steps, to prevent cross collisions in paths between discrete states. An example of the output of our reachability analysis over horizon $H = 5$s for a human is shown in Fig 2.5. The most likely future path, connected

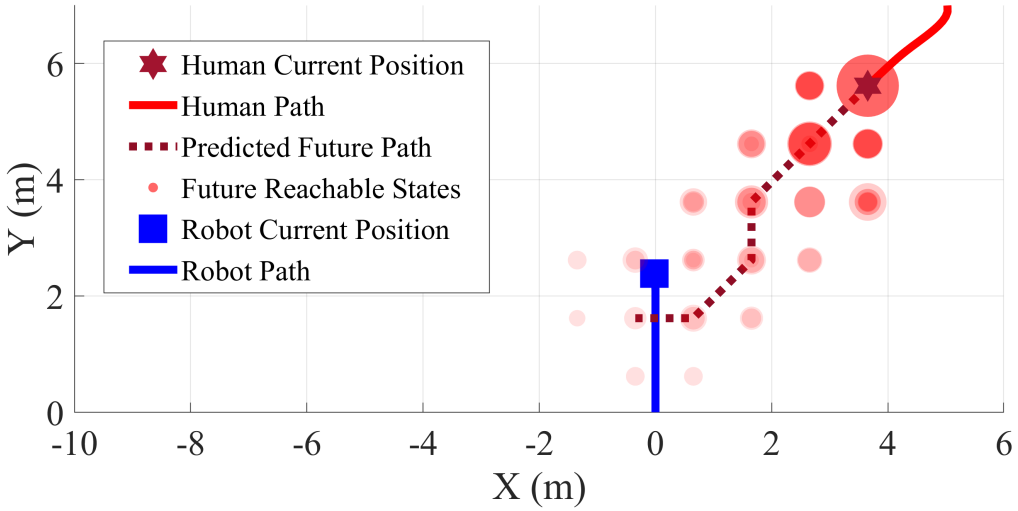


Figure 2.5: Stochastic reachable set of a human in the robot's sensing range. The dotted line shows the most likely future path for the human. Red markers represent reachable states, and the color fades temporally along the horizon, with the lightest at $t + H$. The probability of reachable states is shown by the size of the markers, which decrease as probability decreases.

by the dotted line, is obtained with $\mathcal{P}$ and is nearly consistent with a linear path. Only considering this path, while potentially accurate, provides little advantage over well-known approaches [34, 44], that explicitly assume dynamic obstacles will maintain their current speed and/or direction with minor uncertainty. The stochastic reachable set in our approach leverages the other observations $\mathcal{O}$ and captures uncertainties and irregularities that exist in human motion, addressing the possibility that a human can enter states outside of the most likely path.

2.3.3 Robot Motion Planning

In this section, we present our motion planning technique that takes into account the temporal stochastic reachable sets $\mathcal{R}(t+\tau)$ developed in Section 2.3.2 to proactively avoid and accommodate future motion of surrounding humans. We utilize virtual spring-mass-damper interactions [62] to generate robot motion that avoids and accommodate humans future states while reaching the goal. In our work, virtual springs are built between robots and humans' reachable states at every future

time-step, to drive the robot away from future predicted positions of humans, and at the same time, we build a virtual spring that drives the robot to its goal.

The repulsive spring force directs the robot from its position $\boldsymbol{q}_r(t)$ away from the temporal reachable states $s_j \in \mathcal{R}(t+\tau)$. The springs for each reachable state have constants $k(s_j) = p(s_j)p_e(s_j)$ that depend on the state probabilities output from stochastic reachability analysis, $p(s_j)$, and the crossing probability of the state from the emission matrix $p_e(s_j) \in \mathcal{B}$. The inclusion of crossing probability creates a stronger reaction to a crossing person and vice versa. The extension of a repulsive spring for each reachable state is defined as $d_{ho} = l_h(s_j) - l_o$, where $l_h(s_j) = ||\boldsymbol{q}_r(t) - \boldsymbol{q}(s_j)||$ is the distance from the robot to the position of each temporal reachable state, and l_o is a safe distance to maintain between robot and human.

In crowded situations, even with good predictions, a human may get very close to the robot, compromising the safety of the operation. In such cases, we introduce a fail-safe distance, l_d, subject to $l_d < l_o$, which will produce stronger repulsive forces.

Then, the repulsive spring force at each time $t + \tau$, with $\tau \in [1, H]$, for each state $s_j \in \mathcal{R}(t + \tau)$ is computed as follows:

$$\boldsymbol{u}_{rep,(s_j)}(t+\tau) = \begin{cases} k(s_j) d_{ho} \vec{\boldsymbol{d}}_{\boldsymbol{ho}}, & l_d < l_h(s_j) \leq l_o \\ d_{hd} \vec{\boldsymbol{d}}_{\boldsymbol{hd}}, & l_h((s_j)) \leq l_d \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

where $\vec{\boldsymbol{d}}_{\boldsymbol{h}}$. indicates a unit vector in the direction away from the human's reachable state. Note that probabilities are left out of the case $l_h(\boldsymbol{q}(s_j)) \leq l_d$, and the spring stiffness is simply 1, which is the maximum value $k(s_j)$ can take, creating the strongest repulsive forces.

The attractive force directs the robot from its position $\boldsymbol{q}_r(t + \tau)$ towards the goal, $\boldsymbol{q}_g = [x_g, y_g]^\intercal$, and is computed as follows:

$$\boldsymbol{u}_{att}(t+\tau) = k_{att}(||\boldsymbol{q}_r(t+\tau) - \boldsymbol{q}_g||) \vec{\boldsymbol{d}}_{\boldsymbol{g}} \tag{2.13}$$

where $\vec{\boldsymbol{d}}_g$ is the unit vector directed towards the goal, k_{att} is the spring constant. Here, the distance to goal is used as the extension of the spring, as the ultimate target of the robot is to reach the goal.

The summation of all components (attractive and repulsive forces) yields an input for the robot at each time:

$$\boldsymbol{u}(t+\tau) = \boldsymbol{u}_{att}(t+\tau) + \sum_{j=1}^{n} \boldsymbol{u}_{rep,\boldsymbol{q}(s_j)}(t+\tau) - c_d v(t) \tag{2.14}$$

where n is the number of states reachable at $t+\tau$ and $c_d v(t)$ is the spring damping effect. At each time-step, the input is used to compute the robot's next position, and the entire procedure is repeated for the remainder of the horizon, resulting in a time series of inputs for the robot.

Instead of applying the first input in the predicted series (reactive approach), we proactively replan the trajectory to accommodate the future states by finding the first time τ^* in which the robot deviates from a direct path (i.e., a straight line) to the goal,

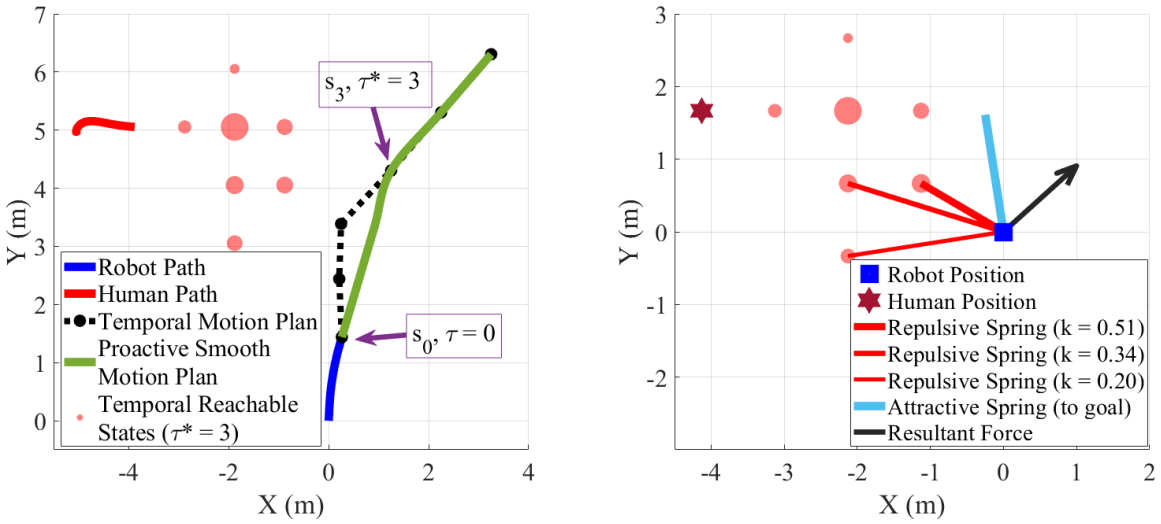
$$\begin{aligned} \tau^* &= \arg\min_{\tau}(\boldsymbol{u}(t+\tau)), \\ \text{s.t. } \boldsymbol{u}(t+\tau) &\neq \boldsymbol{u}_{att}(t+\tau) \end{aligned} \tag{2.15}$$

Then, we compute a new set of inputs that directly sends the vehicle towards the planned position at τ^*, accommodating the deviation caused by future states of surrounding humans:

$$\boldsymbol{u}'(t+\tau) = \frac{\sum_{\tau=1}^{\tau^*} \boldsymbol{u}(t+\tau)}{\tau^*},\ \tau \in [1, \tau^*] \tag{2.16}$$

where the numerator is the vector sum of the inputs between 1 and τ^*. Dividing this resultant by τ^* provides the value of the inputs to use to move directly toward the deviated position at τ^*. When $\tau^* < H$, we include the previously calculated inputs, $\boldsymbol{u}(t+\tau),\ \tau \in (\tau^*, H]$, to populate the complete series of inputs for the horizon, which is then smoothed with cubic spline interpolation [12]. The final smoothed trajectory is sent to the robot, and is replanned at every time-step, as the presence and motion of surrounding humans is constantly changing and evolving. Fig. 2.6(a) shows an example of such motion plan. The robot is in state s_0 at $\tau = 0$ and wants to reach a goal g. After running the prediction, a deviation from the planned straight line trajectory occurs at $\tau^* = 3$ resulting in s_3. With our procedure, the robot replans its trajectory to go directly to s_3 from s_0. Fig.2.6(b) shows the predicted action (black arrow computed with (2.14)) of the robot as a

resultant of the repulsive spring forces (red lines) and attractive force (light blue line) pushing the robot away toward the right side of its desired trajectory. Note that, per (2.12), only the reachable states within the range l_o impact the robot.



(a) Overall plan of the robot at $\tau = 0$ after predicting a deviation at $\tau^* = 3$.

(b) The springs formed with the predicted human positions at $\tau^* = 3$ and the resultant motion from the local frame of the robot.

Figure 2.6: Motion plan prediction example.

In our approach, taking into account the probabilities of reachable states and emission probabilities prevents the robot from reacting too much to an unlikely state or non-crossing state, while ensuring that a more likely or crossing state creates a stronger reaction from the robot. Considering only temporal reachable sets to plan at each time provides an advantage over a potentially circuitous or "frozen" paths generated by approaches that consider the entire range of future positions at one time [34]. In addition, replanning inputs based on future deviation generates robot behavior that accommodates humans' future intentions, creating an advantage over reactive dynamic obstacle avoidance approaches that only consider the current positions of surrounding humans.

2.3.4 Online Model Updates

Our motion planner is predictive and proactive with respect to future states of humans, but due to the dynamic nature of the environment, many new behaviors can be observed online. Consider a case, for example, in which a robot observes a new behavior at one point during its operation. If

the predictive model can be updated online, the expectation that the same behavior could occur at a later time can be exploited to improve predictions, and by extension, motion planning.

To this end, we propose online updates of observations and the emission matrix used for prediction and planning. The set of observations $\mathcal{O}$, main input to our stochastic reachability, is updated with a new observed state transition at runtime $o(t) = s_j$. The emission matrix, $\mathcal{B}$, a key part of our motion planning, is updated using the procedure described in (2.5)-(2.7), by incrementing the instances of emissions and occurrences of the state, resulting in an updated $\mathcal{B}'$ that includes behavioral information from the new observations. Because the updated matrix is still bounded by the size of the state space N and emission space M, and the update procedure is an element-wise operation, the worst-case computational complexity cannot exceed $O(NM)$ [71]. This update can occur within one iteration of robot operation, rendering the updated predictive model usable at the next iteration.

Updating and learning online is unique to our approach, as most learning-based techniques, such as DNNs [106], consist of training complex connections between inputs and outputs that cannot necessarily be accessed [27], making them difficult to update without fully retraining the network, which can be a computationally intensive and time consuming procedure. In our approach on the other hand, we understand that emission probabilities are computed by counting the instances of each state and instances of behaviors, allowing us to easily update them as we add observations to $\mathcal{O}$ at runtime.

2.4 Simulation and Experimental Results

The case study investigated consists of a robotic vehicle performing a go-to-goal operation in the presence of moving humans. The robot is expected to predict the intentions of humans, accommodate, and avoid them as it completes its mission.

2.4.1 Simulations

In the following simulation the robot is tasked to move in a 11m by 12m environment from an initial point $(0, 0)$m to a goal at $(0, 12)$m while navigating around two humans moving in unknown trajectories. The simulated robot trajectories and the distance maintained between the robot and

surrounding humans are shown in Fig. 2.7. Specifically, in Figs. 2.7(a) and 2.7(b), we compare our predictive approach before and after model updates respectively: In Fig.2.7(b) the same simulation was run 20 times updating the model after every run following the approach described above demonstrating that over time the behavior improves, becoming smoother if similar behaviors are recorded several times. In Fig. 2.7(c) we show the results for a reactive-based planner, in which the robot is pushed away from the humans once they are in close proximity, resulting in motion away from the goal.

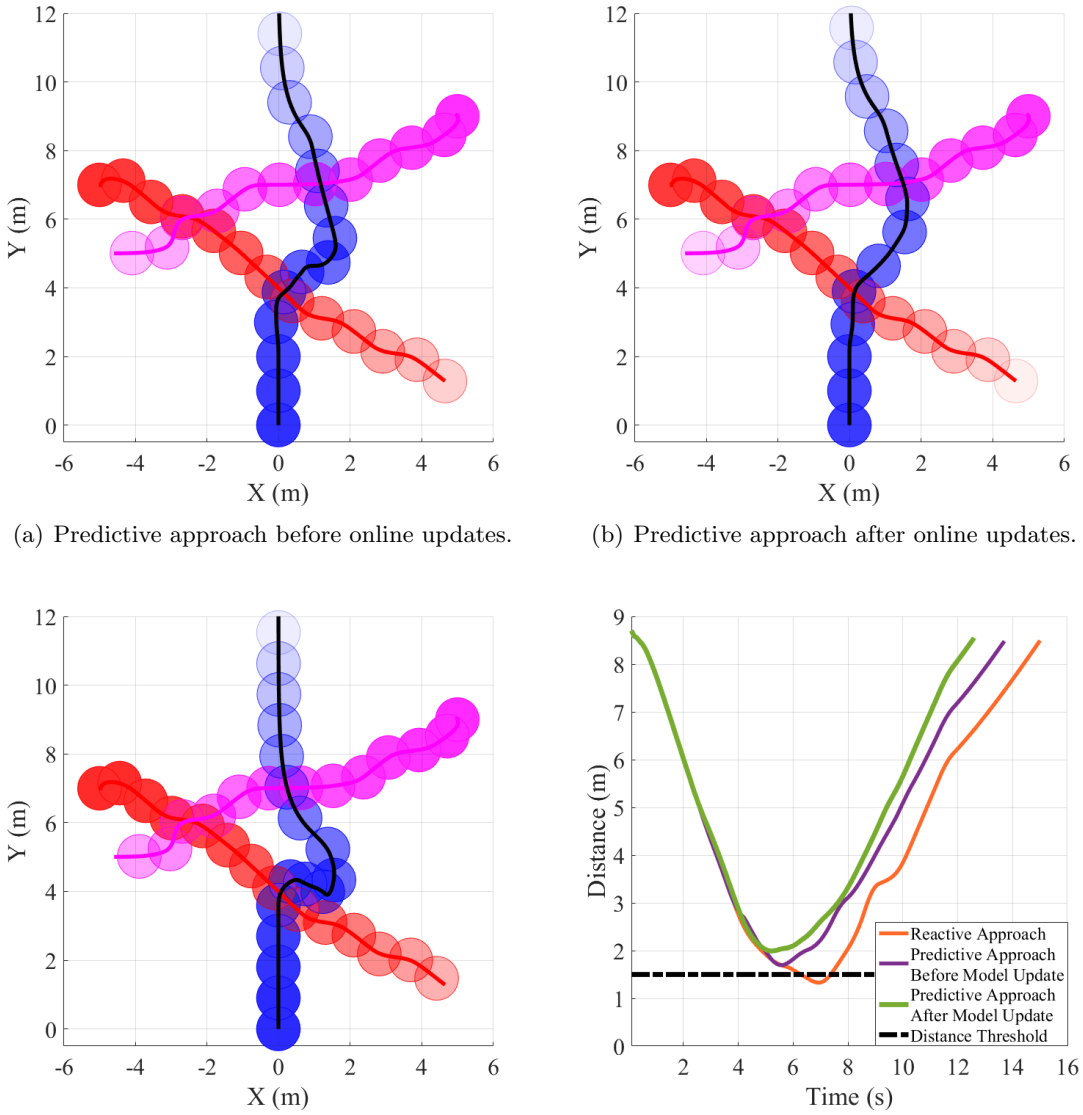
In the simulations, the maximum velocity of the robot, $v_{max} = 1$m/s, was chosen to approximate average human walking speed, which is usually between 0.7m/s and 1.4m/s. The resting length parameter is set to $k_{rest} = 2$, giving $l_o \approx 2$m,, and the fail-safe distance threshold is set to $l_d = 1.5$m. The time horizon for prediction and control was set to $H = 5$s, and the robot observes an area of radius 5m relative to its position.

The online update procedure increases probabilities of previously observed future states, decreasing probability of surrounding reachable states, which in turn, exert weaker repulsive forces (see (2.12)). Thus, the model updates reduce robot interactions with extraneous states. The effectiveness of our approach can also be seen in the added time to goal. The robot time to goal in the presented approach was recorded to $t = 13.7$s before and $t = 12.6$s after model updates, while in the reactive approach, the completion time was $t = 15$s.

We also extensively test our approach on longer trajectories, where the robot traverses a 60m long corridor in the presence of approximately 50 people who walk and stop intermittently. We have also run a comparison with reactive spring-mass-damper planners, and with ORCA [100], a well-known and widely used dynamic obstacle avoidance technique. Comparative results over 100 trials are shown in Table 2.1.

Approach	Added Time (%)	Minimum Distance (m)	Mission Success (%)	Collision (%)
Presented Approach	15%	1.614	96%	0%
Reactive Virtual Springs	34%	1.436	68%	0%
ORCA	18%	1.453	93%	0%

Table 2.1: Comparative simulation results



(a) Predictive approach before online updates.

(b) Predictive approach after online updates.

(c) Reactive approach.

(d) Distance to nearest human.

Figure 2.7: Scenarios with a robot (blue markers, black line) navigating to its goal in the presence of two people (magenta and red markers and lines). The markers fade as time increases and the actors reach their goals. The distance threshold is 1.5m.

The target time for the trajectory is 60s, consistent with a straight path to goal at the maximum velocity, $v_{max} = 1$m/s. Our presented approach adds on average 15%, while the reactive approach and ORCA add 34% and 18% extra time, respectively. The reactive approach often takes circuitous paths, and ORCA is prone to stopping when surrounding humans behave in irregular ways [5], which contributes to the added time. Our approach, on the other hand, predicts the evolution of the scenario and is able to find a path forward, even in the presence of unexpected and irregular

human behaviors.

The presented approach outperforms others in keeping the minimum distance, because the reactive approach only considers current position and ORCA expects other agents (humans) to follow reciprocal velocities, both of which perform poorly when humans take irregular paths with uncertain velocities. For mission success, which we define as reaching the goal within 90s, our approach succeeds in 96% of trials. The instances where our approach does not succeed coincide with those of the other approaches, largely due to a very high density of humans in the corridor, allowing for no safe path forward, due to the safe distance constraints in all three motion planning approaches. Lastly, all three approaches succeed in avoiding collisions, which is the expected outcome.

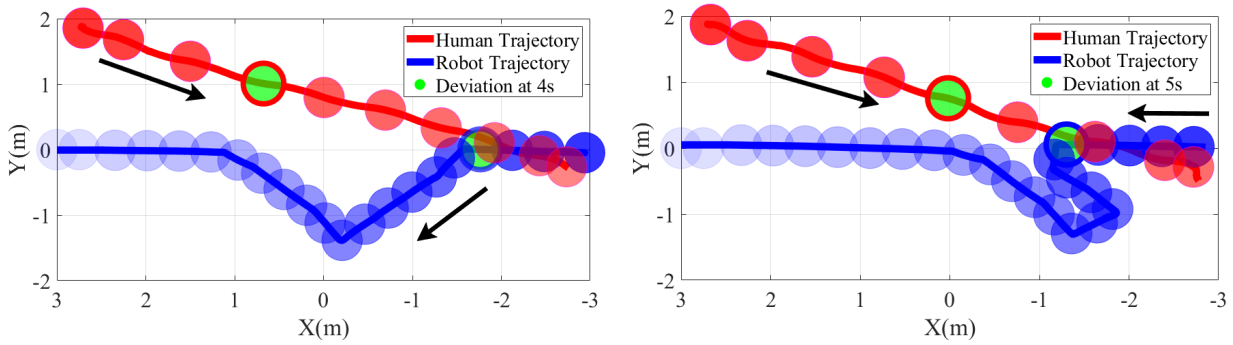
2.4.2 Experiments

The proposed approach was also validated experimentally using a Clearpath Robotics Ridgeback Omnidirectional Platform in an indoor environment. Two kinds of experiments were performed : 1) using our a motion capture system to track the positions of humans and 2) using an on-board ASUS Xtion RGB-D camera with the SPENCER people tracking package [63] to locate humans in the environment. HMM predictions, reachable sets, and the motion plan are computed in MATLAB, and the robot is controlled using the Robotics System Toolbox to interface MATLAB with ROS. The training for both experiments consisted of 100 trajectories depicted in Fig. 2.4(a).

The robot is tasked to avoid and accommodate humans while moving from $(-3.0, 0)$m to $(3.0, 0)$m at a maximum velocity of $v_{max} = 0.5$m/s, which is reduced from our simulations due to Lab space constraints.

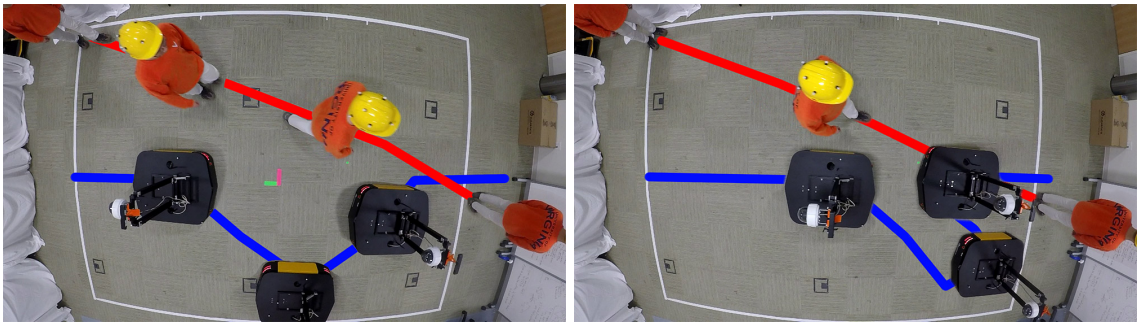
In Fig. 2.8, we show a comparison of the presented proactive approach and a reactive approach. The proactive approach depicted in Figs. 2.8(a,c) predicts the person's future states creating a trajectory to accommodate the person motion while maintaining the direction to the goal. In the reactive case in Figs.2.8(b,d), the robot is pushed backwards by the person moving diagonally. The deviation in the reactive approach occurs when the person is closer to the robot at $t = 5$s.

In the second experiment, we recreate a similar situation as our simulations in Fig. 2.7. The robot behaves similarly and successfully navigates around both humans, accommodating their intentions (Fig. 2.9(a)). Notably, the robot reacts to the purple trajectory by moving in the $+y$ direction, despite the distance never nearing the threshold (Fig. 2.9(b)). These types of experiments



(a) Proactive approach trajectory.

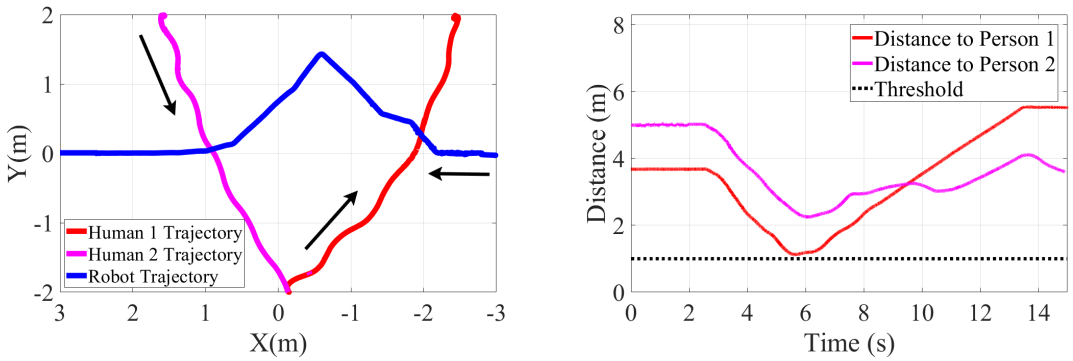
(b) Reactive approach trajectory.



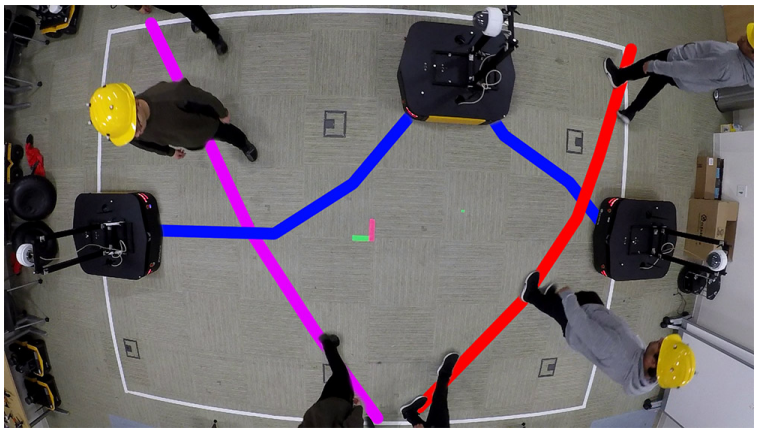
(c) Proactive approach snapshots.

(d) Reactive approach snapshots.

Figure 2.8: Comparative results of the presented proactive approach and a reactive virtual physics based approach. The markers become more transparent as time increases. Green markers represent deviation points and associated times in each case.



(a) (b)

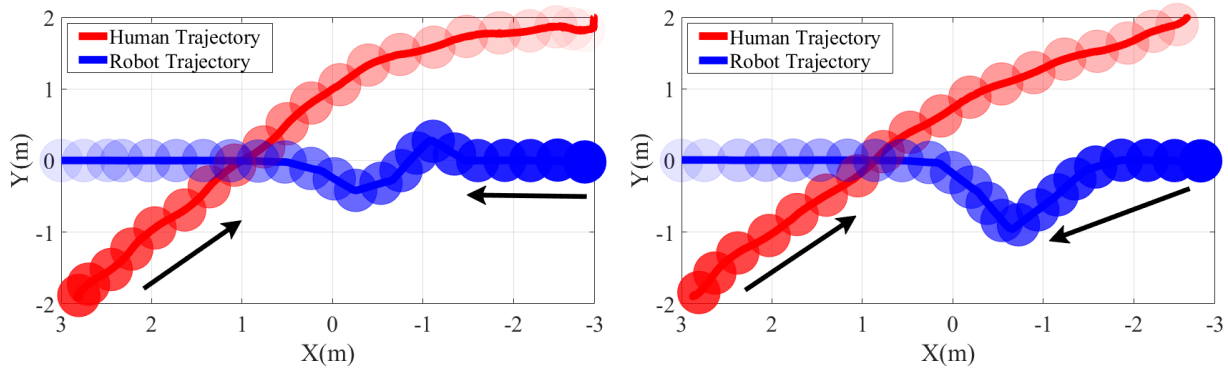


(c)

Figure 2.9: Results from two-person experiments. (a) shows the trajectories of the humans and the robot, while (b) shows the distance maintained between the robot and each person, and (c) shows the overhead snapshots of these experiments.

were carried out with over 30 trajectories in the lab environment, resulting in no collisions and no violations of the safe distance.

To test the ability of our approach to learn from new observations, we perform an experiment with a model that is trained on an incomplete subset of trajectories, comparing before and after the model updated during runtime. Initially, we observe the robot going toward the human's path (Fig. 2.10(a)), as it makes incorrect intention predictions. The person, in this case, takes a slightly wider path, reacting to the robot's incorrect behavior. The model is updated and reinforced with the observed trajectory at runtime, as discussed in Section 2.3.4. When a new similar scenario happens again, the robot correctly predicts the human intention and deviates to accommodate the future path as shown in Fig. 2.10(b).

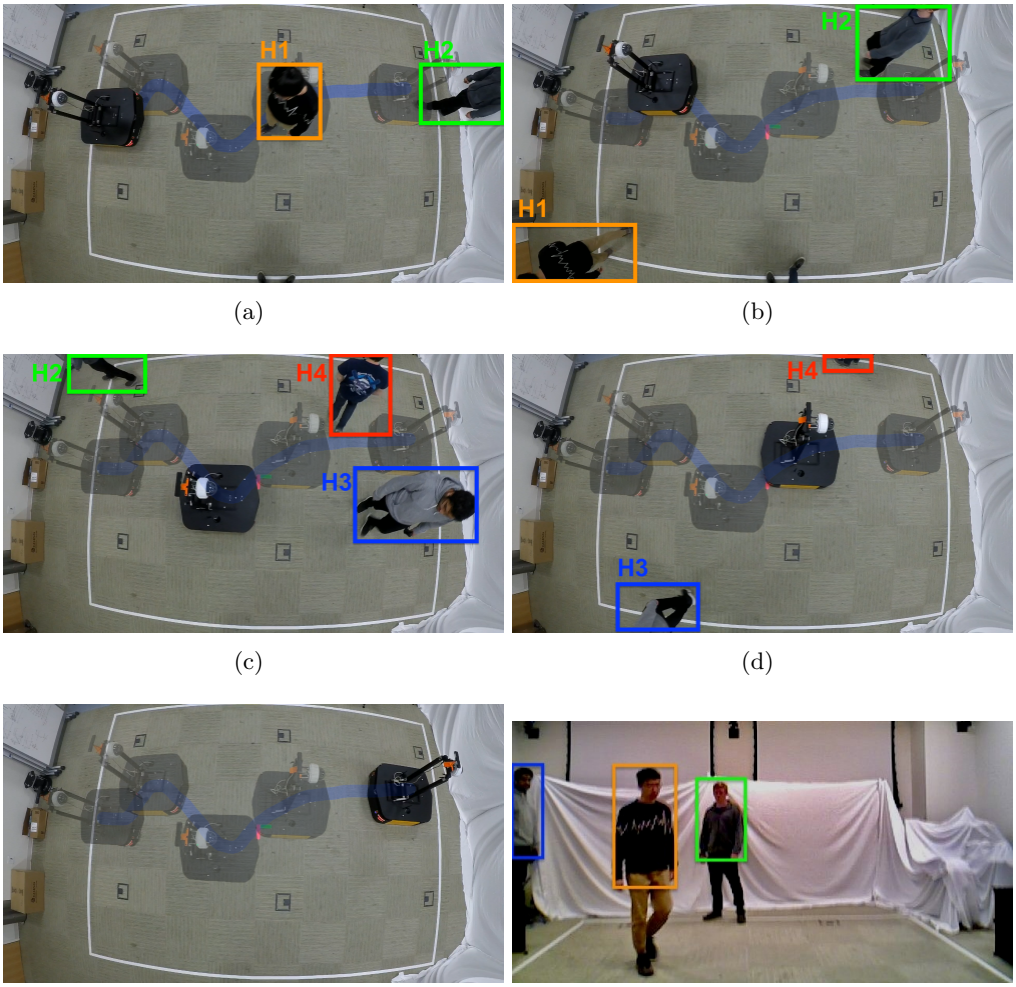


(a) Trajectory prior to model update.

(b) Trajectory after model update.

Figure 2.10: (a) A robot with a poorly-trained model fails at detecting the human intention, moving into the path of the human. However, when observing a similar trajectory, due to the updating process, the robot predicts and accommodates the correct human intention by moving to its left (b).

In the second experiment without MOCAP using only the on-board RGB-D camera, the robot successfully accommodates four people and reaches the goal, showing that our approach scales to more people and performs well despite noisy camera measurements and uncertainty in person identification and tracking. An overlaid sequence of snapshots and a first-person view of the robot are displayed in Fig. 2.11.



(a) (b)

(c) (d)

(e) (f)

Figure 2.11: Results from 4 person camera experiment. Fig 2.11(f) shows a first-person view of the robot when 3 of the 4 people are in the frame.

2.5 Discussion

In this chapter, we have presented an approach for prediction and planning of a robot traversing a shared environment with multiple humans. We leverage Hidden Markov Model theory to predict the human intention and propose temporal stochastic reachability that is coupled with a virtual spring-mass system-based method to generate proactive intention-aware motion for the robot. Results show that our approach performs better than reactive dynamic obstacle avoidance approaches and ORCA. Unique from other works in this field, a key feature here is that the predictive model is constantly updated at runtime improving the behavior as more observations are made. The reason that this happens is because human motion is not random and thus over time, the system is able to learn and converge to common social behaviors.

2.5.1 Limitations and Future Directions

Our approach makes explicit predictions about the future positions of humans and accounts for uncertainties surrounding these predictions. But in cases where there are multiple humans surrounding the robot, it is conceivable that the "freezing robot" problem may appear. Although temporal reachable sets reduce the possibility of the robot to completely stop, and any freezing in our approach would be temporary, these behaviors might be too conservative and not necessary within the flow of a dynamic environment. The uncertainty modeling through stochastic reachability is, however, designed to be conservative, but we note that there may cases in which it is not necessary to explicitly model the future positions of every actor. To deal with these considerations, we leverage classification methods and relax the need to explicitly predict future positions of actors in the environment. These techniques are introduced in the following chapters.

Chapter 3

Interpretable Runtime Prediction and Planning in Co-Robotic Environments

In this chapter, we introduce our interpretable monitoring framework for proactive non-interfering motion planning for autonomous mobile robots that share environments with dynamic actors. This interpretable monitoring framework leverages decision tree (DT) theory to the predict and explain the causes for why a robot may or may not interfere with the motion of nearby dynamic actors. The DT-based monitor serves as a high-level planner for the robot and is paired with a pure-pursuit low-level planner that determines the controls sent to robot. The key difference from end-to-end learning based approaches is that our monitor is able to provide human-readable explanations and reasoning for its decisions. Furthermore, we build lightweight DTs at runtime, enabling us to incorporate new data as the robot observes new behaviors at runtime. We validate this approach with both simulations and experiments with a UGV navigating in the presence of multiple people. This work has been published at the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

3.1 Introduction

Predicting future states of dynamic actors to compute stochastic reachable sets is very powerful in informing how the robot can generate proactive, accommodating motion. However, such explicit predictions may not always be necessary and can be impractical for motion planning in environments

with lots of dynamic actors, in which the robot's predictions may be inaccurate, constantly changing, or too restrictive for robot motion. Another consideration is that typically, dynamic actors in the presence of a moving robot will behave rationally and take upon the burden of avoiding the robot on their own if the robot is not cooperating. An accommodating robot, on the other hand, should proactively minimize creating such a burden on nearby actors.

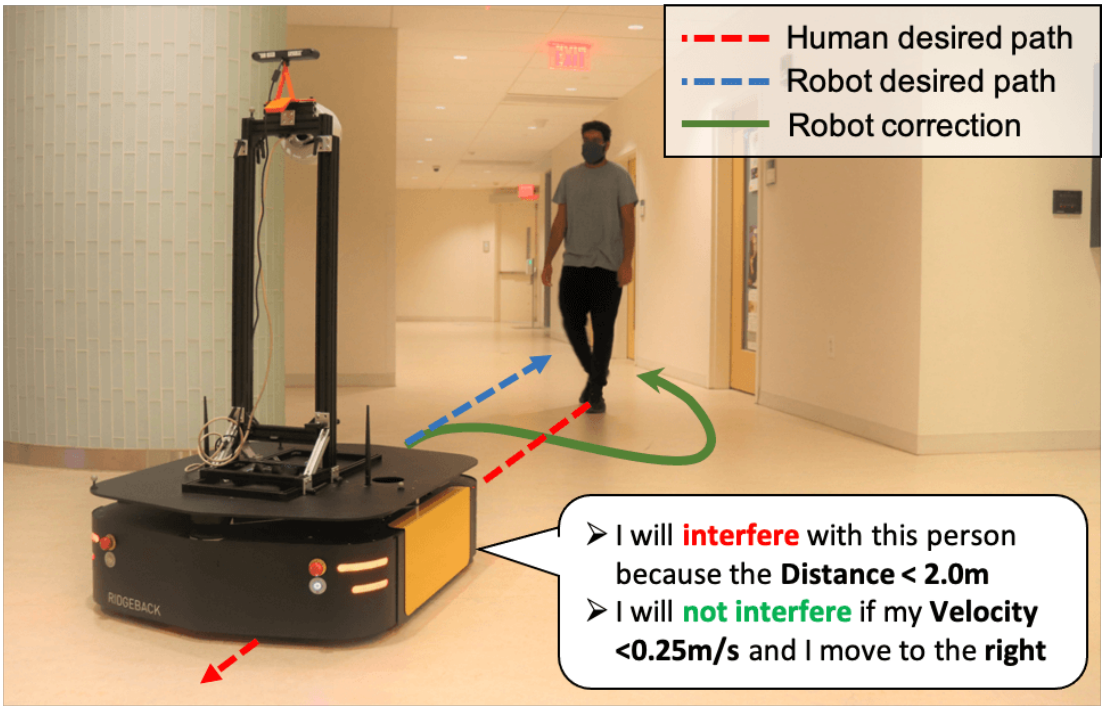


Figure 3.1: In our proposed approach, a robot predicts, explains and finds a corrective action to avoid interfering with an oncoming human.

Humans, contrary to most deployed robots, are able accommodate others in very intuitive and easily interpretable ways without exactly predicting where others will go. We are generally aware of our actions, and we can assess and explain if attributes of our behavior (e.g., how fast we are moving) will lead to some type of *interference* with other people, causing them to change their path [13]. We not only are able to explain whether we are interfering, but also intuitively use this explanation to change our motion, all without making explicit future state predictions of other actors. If robots could reason about their behavior and plan corrective actions in a similar way, their motion would be easy to understand and interpret for surrounding humans.

More recently, advanced machine learning techniques like Long Short Term Memory (LSTM) networks and Deep Reinforcement Learning (DRL) have been used to generate more natural robot

behaviors around humans [47, 31, 106]. While good robot behaviors can be produced, the approaches often contain black-box models [3], and are unable to provide explanations or reasoning for decisions. In addition, these approaches typically are not adaptable at runtime and require dedicated training phases.

With that, the two main challenges we aim to solve in this chapter are the following:

1. How to predict and explain whether a robot will interfere with the path of a nearby moving actor.

2. How to leverage predictions and explanations (i.e., interpretability) to inform the robot how it should correct its behaviors to avoid interfering with dynamic actors.

To solve these challenges and interpretably mitigate robot behaviors that interfere the paths of other dynamic actors, we propose a novel method that leverages decision tree theory [16] to predict, explain, and plan corrective actions at runtime in situations in which a robot will interfere with the motion of surrounding humans. Different from other learning-based approaches, besides the explainability aspect, another contribution of our work is that previously unobserved and misclassified data are considered at runtime through validation criteria to improve and refine predictions and corrective actions in future operations. In Fig. 2.1, we show a motivating example for this work in which a robot is able to predict, explain, and find a correction to avoid interfering with a person's intended path.

3.2 Problem Formulation

Consider a mobile robot tasked to navigate an environment while avoiding other actors, in particular, humans. Without prior knowledge about the intended goal of the surrounding humans, this robot would not be able to predict their path. We note however that humans tend to move in certain way, typically in the direction of the desired goal. Let us define the path followed by a human as $\boldsymbol{q}_i^*$, with $i = 1, \ldots, N_h$ where N_h is the number of humans in sensing range with the robot. With such premises we would like to design a framework for a robot to directly predict *interference* with all the humans in its sensing range and plan its motion accordingly to minimize the deviation of human paths due to its presence along the way. Formally, the problem can be cast as:

Problem 3.1 (Non-Interfering Motion Planning and Control) *Design a policy to predict and explain future interfering interactions between a robot and surrounding humans and to plan corrective actions, $\boldsymbol{u}$, that do not cause human paths to deviate more than a distance δ from the intended trajectory:*

$$||\boldsymbol{q}_h(t) - \boldsymbol{q}_h^*(t)|| \leq \delta, \forall h = 1, \ldots, N_h(t), t \geq 0 \tag{3.1}$$

where $\boldsymbol{q}_h(t)$ and $\boldsymbol{q}_h^*(t)$ are the observed path and intended path of the i^{th} actor at time t respectively.

A correlated problem that we propose to investigate in this chapter is to improve robot behavior over time in response to previous experience. To this end, we create a strategy to validate and update predictions and planning at runtime when undesirable behaviors are observed or when runtime observations are unmodeled in the training data.

3.3 Methodology

Our proposed interpretable monitoring framework follows the architecture in Fig. 3.2.

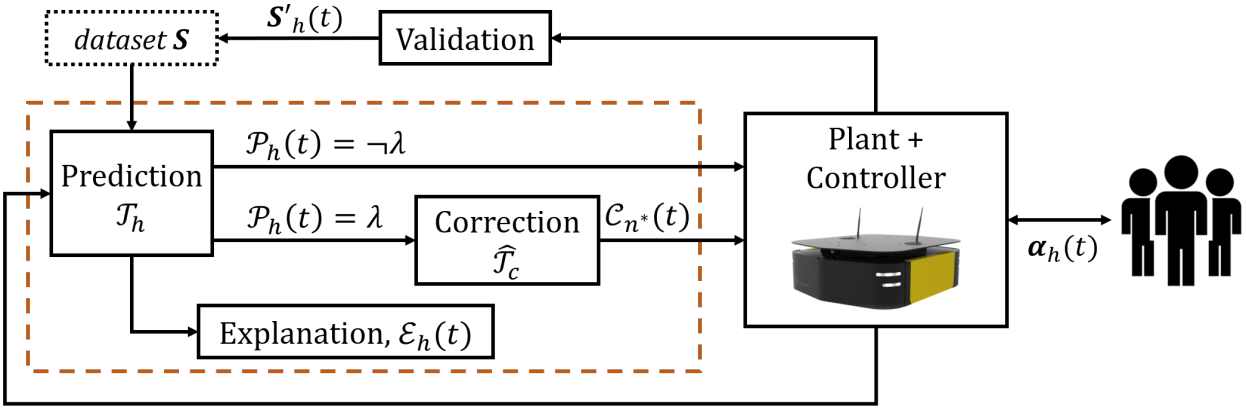


Figure 3.2: Block diagram of the presented approach.

At the core of our framework we leverage decision tree (DT) theory to predict interferences and correct robot behaviors. With observations of surrounding humans $\boldsymbol{\alpha}_h(t)$, a local DT, $\mathcal{T}_h$ is constructed with a dataset of human-robot trajectories to compute a prediction $\mathcal{P}_h(t)$ and explanation $\mathcal{E}_h(t)$ as to whether the robot will *interfere* (λ) or *not interfere* ($\neg\lambda$) with paths of surrounding humans. Then, when interference is predicted, a cascaded (secondary) tree $\hat{\mathcal{T}}_c$ is used to

generate corrective behaviors that reduces robot interference with human paths. Finally, a validation scheme is proposed to update the dataset online, improving future DT operations.

3.3.1 Decision Tree Formulation and Training

Decision trees (DTs) are a form of supervised learning that consist of white-box models which make predictions easy to interpret [16]. DTs are different from most learning-based approaches for classification, such as DNNs [27], which use black-box models and it is difficult to identify the causes that lead to a particular output, or even why that output is appropriate or correct for a given application. In this work, DTs are constructed as binary classification models that are made up of a network of nodes; the outermost nodes, known as leaves, correspond to labels given in the training (decisions). Internal nodes define the split criteria for leaves based on the input variables (attributes). The trees are grown using the Gini Index, which measures the degree or probability of a particular element being wrongly classified when it is randomly chosen. Specifically, given a set of elements labeled with N_c classes, let ρ_i be the fraction of elements labeled with class i, the Gini index is computed as

$$I_G(\rho) = \sum_{i=1}^{N_c} \rho_i \sum_{k \neq i} \rho_k = \sum_{i=1}^{N_c} \rho_i(1-\rho_i) = \sum_{i=1}^{N_c} (\rho_i - {\rho_i}^2) = \sum_{i=1}^{N_c} \rho_i - \sum_{i=1}^{N_c} {\rho_i}^2 = 1 - \sum_{i=1}^{N_c} {\rho_i}^2 \qquad (3.2)$$

For training, we generate a dataset of trajectories in both simulation and in real experiments that consist of human motion from multiple initial to final positions with varying velocity in the presence of a moving robot. In the training, the robot does not react to the humans, so that the prediction model can learn when an interference occurs. In simulation, humans are controlled by a virtual physics-based method [78], which triggers a reaction (interference) if a distance threshold, δ_{th}, is violated. By training in this way, the desired effect is that the robot plans actions that keep a minimum distance of δ_{th} from all surrounding humans.

Attributes should be meaningful to the application, and typically more attributes improve the precision of the prediction. However, too many attributes can lead to redundancy and poor classification [16]. For the human-robot interaction case in this work, attributes are derived from the joint state between the robot and surrounding human, based on explicit sensor data available and implicit data we can compute (e.g., velocity from range sensor readings over time). Specifically,

we define the attributes as

$$\boldsymbol{\alpha} = [\ d_x \quad d_y \quad \theta \quad d \quad d' \quad v_h \quad v_r \quad \ell\ ] \tag{3.3}$$

where d_x, d_y, and θ, are relative x-y positions and heading, respectively, d and d' are the Euclidean distance and distance derivative (i.e., the rate of change of the Euclidean distance) between human and robot, and v_h and v_r are the human and robot velocities, respectively. The robot's operating "lane," ℓ is a discretization of the robot's y-position, assuming that the robot is typically moving forward, i.e., along the x-direction.

The attributes in $\boldsymbol{\alpha}$ were selected via experimental sensitivity analysis and were validated by testing for collinearity, which has shown to affect the performance of classification learners [39]. The correlation matrix in Fig. 3.3 shows no strong linear correlations between any of the attributes, indicating that they are not redundant. To further validate that $\boldsymbol{\alpha}$ is meaningful, we also analyzed

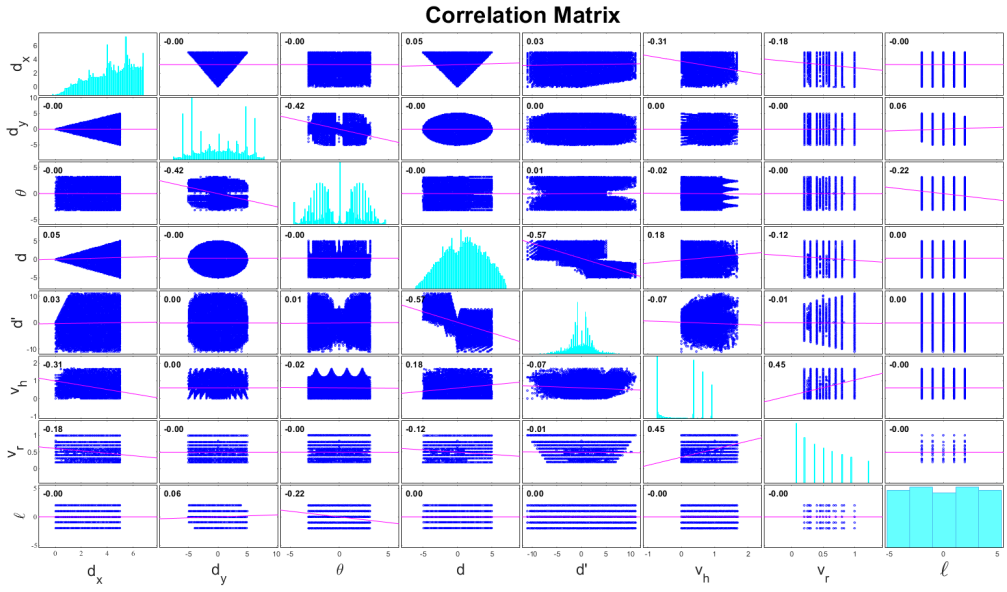


Figure 3.3: Correlation matrix plot of attributes $\boldsymbol{\alpha}$ from simulation data.

the average predictor importance [16] of the attributes over 100 random local trees taken from subsets of the training data. A non-zero importance indicates that the attribute is valuable to decision tree predictions, and it is clear in Fig. 3.4 that while some attributes may be more important than others, all attributes have an effect on the prediction.

Through the training, we obtain a global dataset $\boldsymbol{\mathcal{S}} = \langle \boldsymbol{\alpha_s},\ \boldsymbol{\lambda_s} \rangle$ that includes both the attributes

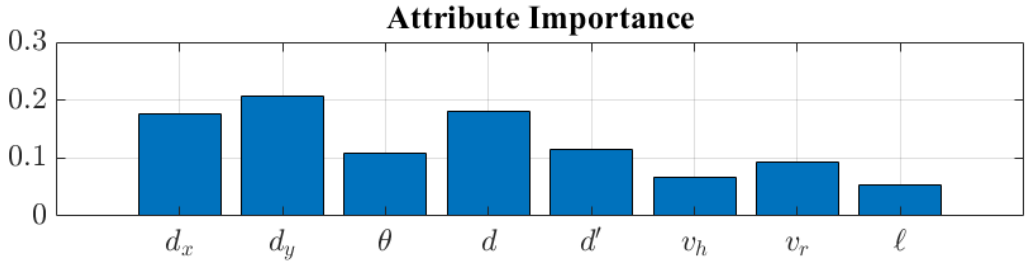


Figure 3.4: Attribute importance as a proportion of total attribute importance.

and corresponding classes of all training instances. Depicted in Fig. 3.5 is the set of experiment training trajectories. There are 21 human paths, and the robot was run with velocities $v_t = [0.2, 0.4, 0.6]$m/s, and on lanes $\ell_t = [-1, 0, 1]$m, giving a total of 189 trajectories. This training dataset is included as "dtree_dataset.7z" in the GitHub repository in [77]. Simulation training was performed similarly with more trajectories, velocities, and lanes.

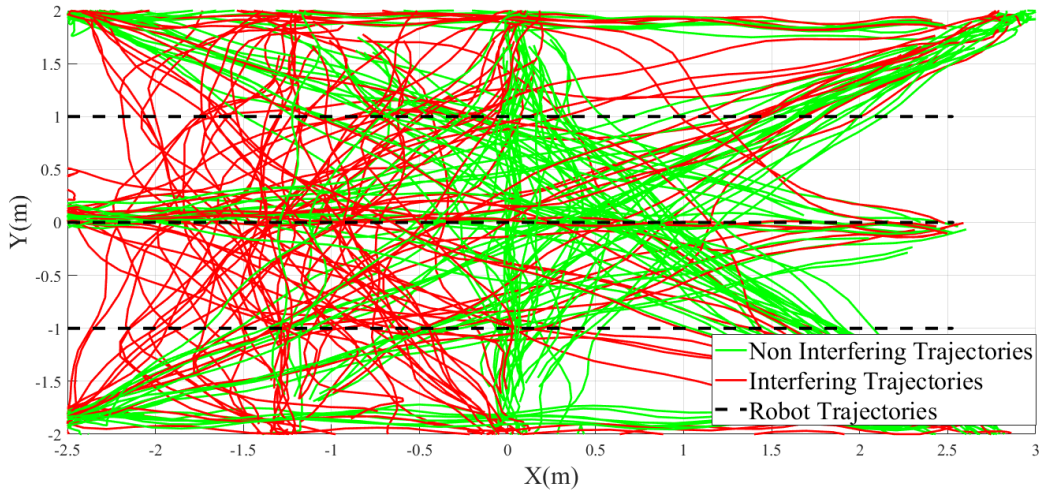


Figure 3.5: Training trajectories recorded in a lab environment with a VICON Motion Capture System.

3.3.2 Prediction and Explanation

Let us now consider first the case of one human approaching the robot. To predict interference, we construct local DTs at runtime with the observed attributes related to a surrounding human, $\boldsymbol{\alpha}_h(t)$, since a global DT for the entire training set can often return inaccurate and imprecise predictions and explanations due to the presence of irrelevant data.

A local tree, $\mathcal{T}_h$, is trained by collecting a subset of points $\boldsymbol{\mathcal{S}_h(t)} \subset \boldsymbol{\mathcal{S}}$ from the global dataset that are within a neighborhood Δ of the attributes of $\boldsymbol{\alpha}_h(t)$:

$$\boldsymbol{\mathcal{S}_h(t)} \subset \boldsymbol{\mathcal{S}} \mid ||\boldsymbol{\alpha}_h(t) - \boldsymbol{\alpha_s}|| \leq \Delta \;\; \forall \boldsymbol{\alpha_s} \in \boldsymbol{\mathcal{S}} \tag{3.4}$$

The distance Δ is a measure of how close the local training data should be to the observed data. The exact value of Δ is selected based on the quality of the available training dataset. With a very rich dataset, a small Δ may result in very accurate predictions and explanations. For a sparse dataset, Δ should be large enough to ensure the decision tree has enough context to generate accurate predictions and explanations. At the same time, in tuning Δ, one should consider that a very large value would provide little benefit over a global tree, while a very low Δ may result in too small a local dataset, removing relevant data. After constructing the local tree, the prediction $\mathcal{P}_h(t) \in [\lambda, \neg\lambda]$ is obtained by evaluating the runtime observation $\boldsymbol{\alpha}_h(t)$ in the tree: $\mathcal{P}_h(t) = \mathcal{T}_h(\boldsymbol{\alpha}_h(t))$.

Generally, a local tree is desirable for prediction and explanation, because a global tree that is constructed with the entire dataset can be detrimental to reasoning about the causes that lead to the predicted state. Intuitively, the robot should only compare a human moving with a heading θ, for example, with training data that reflect similar behaviors; i.e. the heading should not vary too much between local training points and the runtime observation. Another benefit to leveraging local data is that predictions and explanations from a local set of points will be more compact and easier to understand. When local data is used, the number of data points is also reduced, which decreases computational complexity– necessary for building new trees online.

Given a prediction, we compute an explanation $\mathcal{E}_h(t)$ by traversing the path through $\mathcal{T}_h$. A prediction directly corresponds to a leaf, $\mathcal{V}_p$, within the tree. If $\mathcal{V}_0$ is the root of $\mathcal{T}_h$, an explanation is computed by traversing a path, Γ from $\mathcal{V}_0$ to $\mathcal{V}_p$, taking into account the split criterion, c, for the N_i internal nodes along the path. The conjunction of split criterion along Γ is the explanation of the prediction:

$$\mathcal{E}_h(t) = \bigwedge_{k=1}^{N_i} c_k \quad \text{with} \quad \Gamma \mid \mathcal{P}_h(t) \tag{3.5}$$

Traversing all other paths, $\Gamma_j \in \boldsymbol{q}$ with $j = 1, \ldots, N_{\boldsymbol{q}}$ that lead to the opposite decision, $\neg\mathcal{P}_h(t)$, in a similar way, provides a set of counterfactual rules, $\boldsymbol{\mathcal{C}}_h(t)$, to the previously obtained prediction:

$$\boldsymbol{\mathcal{C}}_h(t) = \bigvee_{j=1}^{N_{\boldsymbol{q}}} \bigwedge_{k=1}^{N_j} c_k \quad \text{with} \quad \Gamma_j \mid \neg\mathcal{P}_h(t) \tag{3.6}$$

where each path Γ_j contains N_j nodes to the leaf. These counterfactuals denote which attributes, if changed, would reverse the decision.

Shown in Fig. 3.6 is an example of a local DT used for prediction and explanation based on the following attributes,

$$\boldsymbol{\alpha}_h(t) = \begin{bmatrix} d_x & d_y & \theta & d & d' & v_h & v_r & \ell \\ 1.43 & -4.71 & -81 & 4.93 & -0.43 & 1.0 & 0.6 & 0 \end{bmatrix}$$

The output of the DT is $\mathcal{P}_h(t) = \lambda$, shown by the dark red path and leaf node. Through (3.5) we compute an explanation:

$\mathcal{E}_h(t) =$ ***Interfering*** because: $\{d' < 0.24, d_x < 1.76\}$

Through (3.6), we compute the following counterfactuals:

$\boldsymbol{\mathcal{C}}_h(t) =$ ***Not Interfering*** when:

$\{d' > 0.24, d_x < 1.15\} \vee$

$\{\theta > -56, d' > 0.24, d_x < 1.15\} \vee$

$\{-0.45 \leq d' < 0.24, 1.76 \leq d_x < 1.83\} \vee$

$\{-0.58 \leq d' < 0.24, d_x > 1.83, d < 4.65\}$

The point at which this prediction is made is highlighted in Fig. 3.7, taken from our MATLAB simulations. The prediction is made as soon as the human is within sensing range (5m in this case) of the robot.

As seen in the example, DTs used for prediction provide a logical and interpretable explanation, but the counterfactuals cannot be controlled by the robot's actions alone, as they pertain to human-dependent attributes.

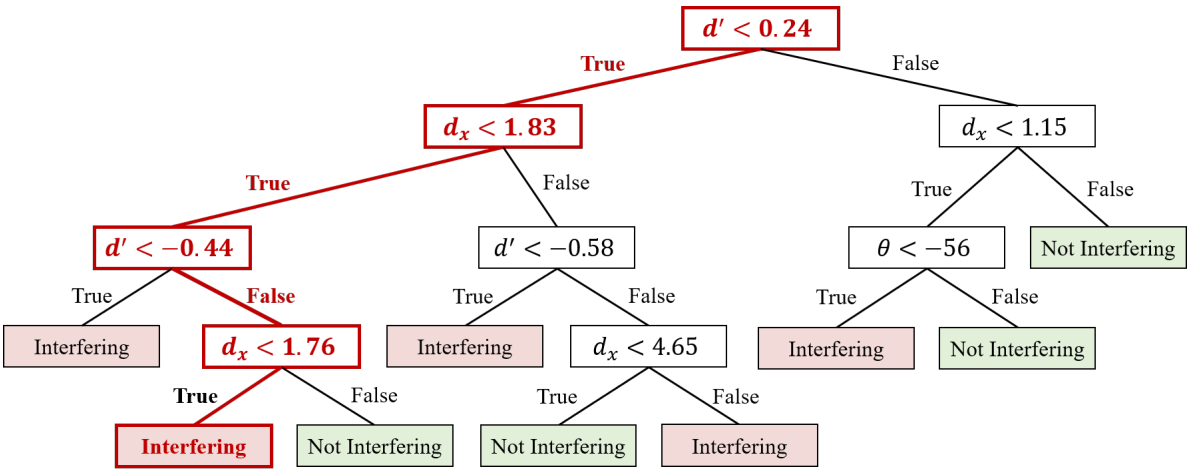


Figure 3.6: Example prediction DT. The internal nodes (white squares) of the tree are binary tests on one of the attributes and the leaf nodes (colored squares) are the class decisions. The bold path shows the current decision.

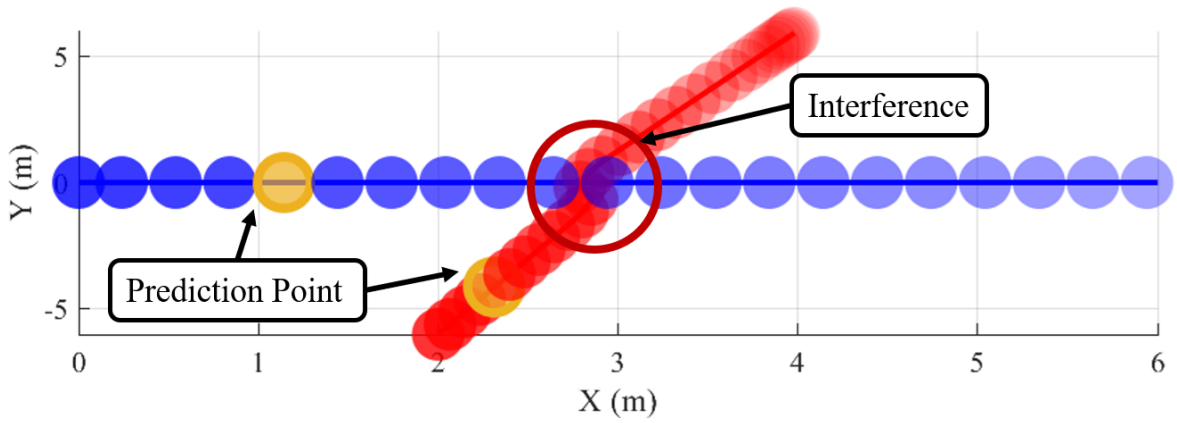


Figure 3.7: Human (red) and robot (blue) trajectories, showing the point (in yellow) at which a prediction is made. The markers fade as time increases and the actors reach their goals.

3.3.3 Corrective Counterfactual Analysis

In case of interference, the counterfactuals provide a set of configurations in which the robot would not have interfered with the path of the human. However, it is not practical to manipulate attributes like distance derivative d' or relative heading θ, since they depend on human motion.

The only controllable attributes in our case are the velocity and lane to track by the robot, $\boldsymbol{\alpha}_r = [v_r,\ \ell]$. To generate actionable counterfactuals, we build a secondary tree, $\mathcal{T}_c$ in which we first fix the human dependent attributes $\boldsymbol{\alpha}_c = \boldsymbol{\alpha}_h \setminus \boldsymbol{\alpha}_r$ and search in the training set for similar attributes as done in (3.4) creating a new set $\boldsymbol{\mathcal{S}}_c \subset \boldsymbol{\mathcal{S}}$ (note that $\boldsymbol{\mathcal{S}}_h \subset \boldsymbol{\mathcal{S}}_c$). In this way, the new DT remains local in the human-related attributes but includes different lanes and velocity pairs, enabling the system to find suitable corrections. The new DT output in this way will be decisions and counterfactuals that only include v_r and ℓ.

In Fig. 3.8, we show the correction tree associated with the example in Section 3.3.2 and Fig. 3.6 in which the robot was initially running with $v_r = 0.6$ and $\ell = 0$. After running the procedure in

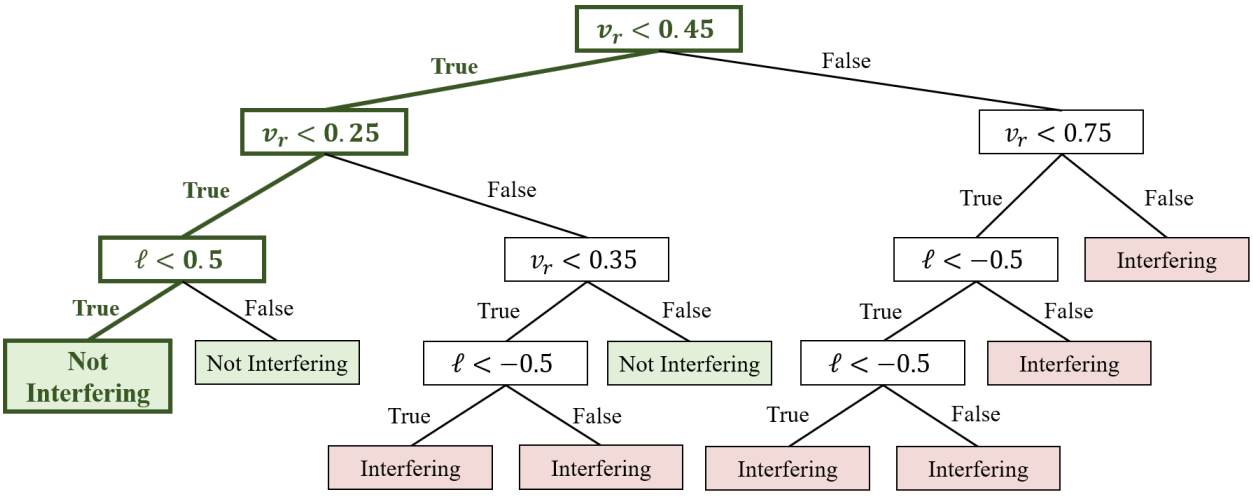


Figure 3.8: Example correction decision tree. Nodes and split criteria only pertain to $[v_r,\ \ell]$. The bold path shows the optimal counterfactual $\mathcal{C}_{n^*}(t)$.

this section, with (3.6), we obtain the following set of actionable counterfactuals:

$\boldsymbol{\mathcal{C}}_h(t) =$ ***Not Interfering*** when:

$$\boldsymbol{\{v_r < 0.25, \ell < 0.5\}} \vee$$

$$\{v_r < 0.25, \ell \geq 0.5\} \vee$$

$\{0.35 \leq v_r < 0.45\}$

To decide which counterfactual to select, we consider two measures that describe the quality of the nodes of the tree, *node error*, e_n, and *node risk*, r_n, with $n = 1, \ldots, N_c(t)$, where $N_c(t)$ is the number of counterfactuals. Node error is the fraction of differently classified training points at a specific leaf. For a leaf that predicts $\neg\lambda$, the node error is:

$$e_n = 1 - p(\lambda) \tag{3.7}$$

Node risk is a weighted measure of impurity (Gini Index in our work):

$$r_n = 1 - \sum_{i=1}^{2} {\rho_i}^2 \tag{3.8}$$

where ρ_i is the fraction of elements labeled with class $i = [\lambda, \neg\lambda]$. A lower node risk indicates that there will be less of chance of an incorrect decision. In our approach, we combine both measures by taking the product $e_n r_n$, since our goal is to identify the best node to use. The use of the product is viable here because both node error and node risk represent different probabilities that rely on information about the training points for the local tree, enabling the use of the general multiplication rule of probabilities for identifying the best node [105]. Then, the optimal counterfactual is computed as follows:

$$n^* = \arg\min_n (e_n r_n) \tag{3.9}$$

The selected counterfactual rule, $\mathcal{C}_{n^*}(t)$ consists of an optimal velocity and lane $\boldsymbol{\alpha^*} = [v_r^*, \ell^*]$. In the example shown in Fig 3.8, $\boldsymbol{\alpha^*} = \boldsymbol{\{v_r < 0.25, \ell < 0.5\}}$.

3.3.4 Corrective Planning and Control

Once a counterfactual rule is selected, the robot moves to implement the correction, which is represented as a "ghost" moving target, and the robot switches into a pure-pursuit based mode of operation [46] until it reaches the ghost vehicle. This is necessary because the corrective action represents what the robot should have been doing at the instance t at which the correction was found, meaning that the robot would only satisfy non-interfering conditions if $v_r(t) = v_r^*$ and $\ell(t) = \ell^*$.

During the pure-pursuit corrective operation, the robot uses the ghost vehicle's state to make predictions until the tracking error between robot and ghost $e(t) = \boldsymbol{p}_g(t) - \boldsymbol{p}_r(t) \approx 0$, after which it reverts to performing predictions based on the actual state of the robot. This is needed because as the robot performs corrective behaviors, we observe transition states that are unmodeled in the training data, since the robot does not react to the humans in the training. In general, for mobile robotic applications as the ones discussed in this thesis, we can assume that transition times are negligible and the robot can quickly reach the ghost vehicle.

3.3.5 Multiple Decision Trees

In this section, we discuss how our approach extends to scenarios with multiple actors. Predictions and explanations can be computed as discussed previously, as the system needs to understand if it's interfering with each actor individually. Finding corrections, however, is more challenging, as the corrective action must not only remove interference with one actor, but also should not cause interference with others. This requires a method to consider multiple DTs at once, taking into account different counterfactual rules.

To consider different counterfactual rules of multiple DTs at once, we use *decision tree ensemble models* (DTEM). The main principle behind DTEM (Fig. 3.9) is that a group of weak learners come together to form a strong learner. Some popular methods for generating DTEMs include bagging,

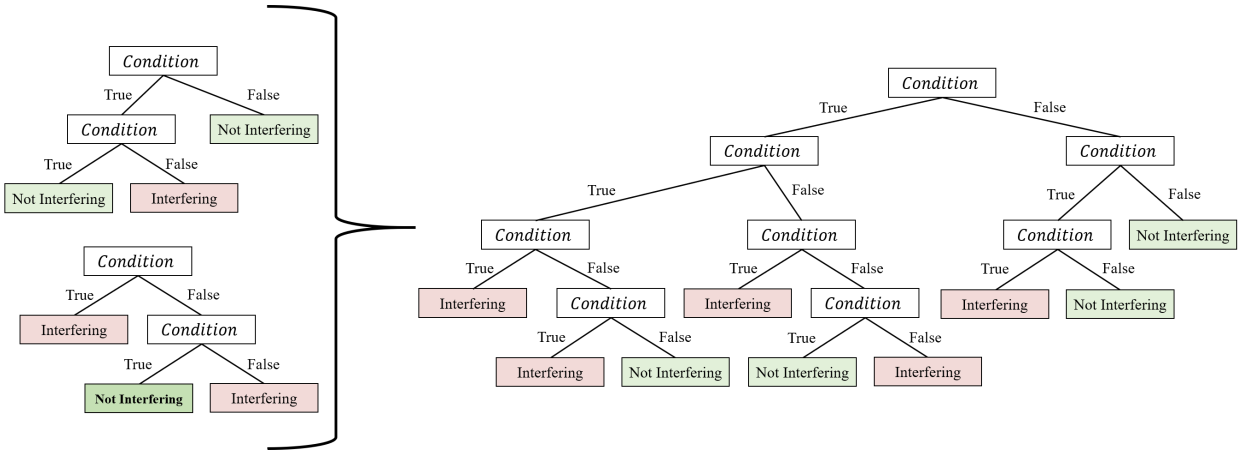


Figure 3.9: General example of a Decision Tree Ensemble Model. Two weak learners come together to form a stronger learner.

boosting, or using random forests, but these consist of random sampling methods [9], which are not viable for our case, since we need to capture all relevant data.

We instead take principles of majority voting DTEMs [4], which typically compare predictions from each DT, and select the majority output. We extend this concept by considering a single combined corrective tree, $\hat{\mathcal{T}}_c$, that incorporates the local training data of all individual trees, rather than just the prediction.

Before combining, the local data for each person are normalized such that $|\boldsymbol{\mathcal{S}}_i(t)| \approx |\boldsymbol{\mathcal{S}}_j(t)|$ with $i, j = 1, \ldots, N_h$, where $|\cdot|$ gives the size of the enclosed dataset. In this way, corrections will not be incorrectly biased towards those with more local data. We combine to obtain the training dataset: $\hat{\boldsymbol{\mathcal{S}}}(t) = [\boldsymbol{\mathcal{S}}_1(t), \ldots, \boldsymbol{\mathcal{S}}_{N_h}(t)]$, with which $\hat{\mathcal{T}}_c$ is constructed and counterfactuals are analyzed with the procedure discussed in Sec. 3.3.3, to obtain the optimal target, $\boldsymbol{\alpha}^*$, and the robot is controlled as described in Sec. 3.3.4.

Building $\hat{\mathcal{T}}_c$ in this way, however, only enables the system to find the best action if one exists within the data, meaning that there can be cases where all considered corrective actions are interfering. This can happen if: 1) the considered corrective data are sparse, meaning that the training set is not rich enough or 2) all possible actions are not feasible, for example, if the robot is surrounded by a crowd. For the former case, we propose a randomized approach to choose a velocity-lane pair that is not included in the local data: $\boldsymbol{\alpha}_s \setminus \hat{\boldsymbol{\alpha}}(t) \in \hat{\boldsymbol{\mathcal{S}}}(t)$. By doing so, the vehicle explores new options until it finds a solution. If no solution is found, the robot switches into a fail-safe mode of operation, which consists of a very low velocity and a reactive obstacle avoidance behavior for safety.

3.3.6 Online Validation and Updating

Due to the dynamic and dense nature of the environment, new and unmodeled human behaviors can be observed and corrective actions may not always eliminate all interference. Since our DTs are constructed at runtime, it is possible to introduce new data to the training set $\boldsymbol{\mathcal{S}}$ as observations are made. To avoid an exploding dataset, we introduce two test cases for adding new data: 1) decision validation, and 2) checking for unbounded observations.

Case 1 Decision validation is necessary when corrective actions from $\hat{\mathcal{T}}_c$ still result in an interference. This can occur when observations contain attribute values that are close to splitting

conditions in the DTs leading to misclassification, or when a correction cannot be found within the DTEM. If the robot observes that the distance threshold δ_{th} is violated at runtime, the recorded attributes are included in $\boldsymbol{\mathcal{S}}$.

Case 2 If an observation is outside the bounds of the training data, reliable predictions or corrections cannot be expected. It has been shown for learning components that testing data within the vertices of the smallest convex set around training data produces the most accurate predictions [26]. In this work, convex hulls are generated around local training data using the Quickhull algorithm [93] to form a boundary denoted $Conv(\hat{\boldsymbol{\mathcal{S}}})$. Then, we check if observations are within the outermost points of each dimension (attribute) of the convex hull using linear inequalities [6]:

$$\min(Conv(\hat{\boldsymbol{\mathcal{S}}})) \leq \boldsymbol{\alpha}_h(t) \leq \max(Conv(\hat{\boldsymbol{\mathcal{S}}})) \tag{3.10}$$

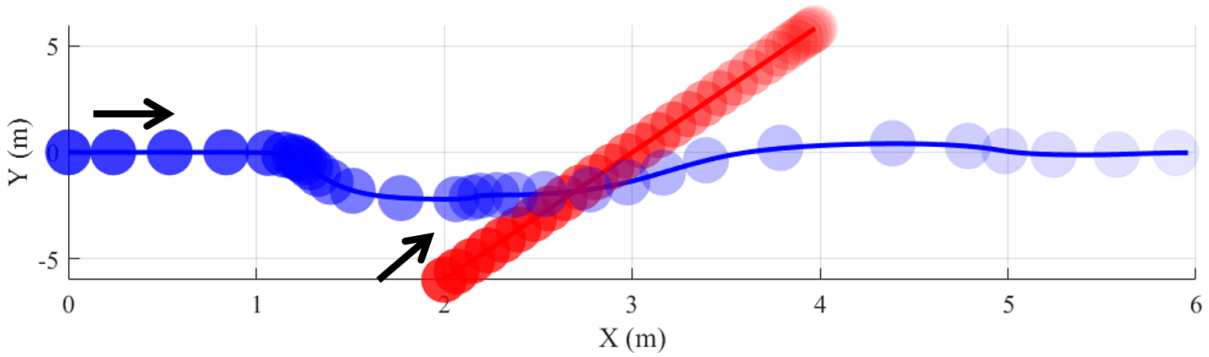
If any part of $\boldsymbol{\alpha}_h(t)$ is outside the convex hull, the data are labeled and included in $\boldsymbol{\mathcal{S}}$. The dataset updates at runtime through the presented test cases, resulting in more refined local trees, and therefore better decision-making in the future.

3.4 Simulations

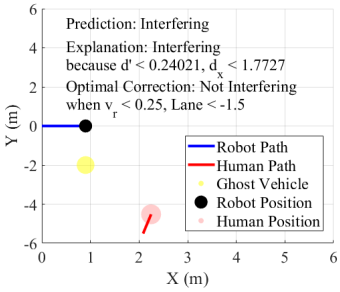
We performed a series of simulations in MATLAB to test the effectiveness of our approach. Training included velocities $\boldsymbol{v_r} = [0.2, 0.4, 0.6, 0.8, 1.0]$m/s, and lanes $\boldsymbol{\ell} = [-2, -1, 0, 1, 2]$m simulating a classical non-holonomic UGV. The robot considers humans within a range of 5m and has a nominal velocity of 0.6m/s, and corrections are limited to any discrete velocity or lane seen in the training, that is from $\boldsymbol{v_r}$ and $\boldsymbol{\ell}$, respectively.

In the baseline simulation shown in Fig. 3.10, the robot (blue) is tasked to move from $(0, 0)$m to $(6, 0)$m while predicting, explaining, and correcting to avoid interfering with a person (red) moving along a trajectory previously unknown to the robot (i.e., different from the training set).

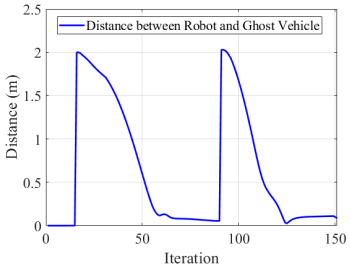
The paths are shown in Fig. 3.10(a), and prediction, explanation, optimal correction, and ghost vehicle are shown in Fig. 3.10(b). The robot predicts $\mathcal{P}_h(t) = \lambda$ and determines that it must apply the correction: $\boldsymbol{\alpha}^* = [v_r < 0.25, \ell < -1.5]$. The ghost vehicle (yellow marker) immediately applies these corrections. In Fig. 3.10(d), the distance between robot and human is shown to verify that the distance threshold $\delta_{th} = 1.5$m is not violated.



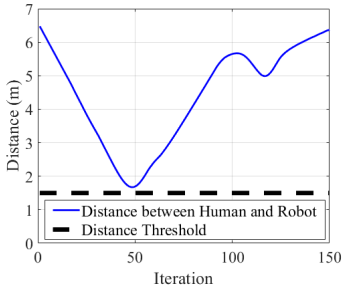
(a) Paths of human(red) and robot(blue).



(b) Decision tree operations.



(c) Distance to ghost vehicle.

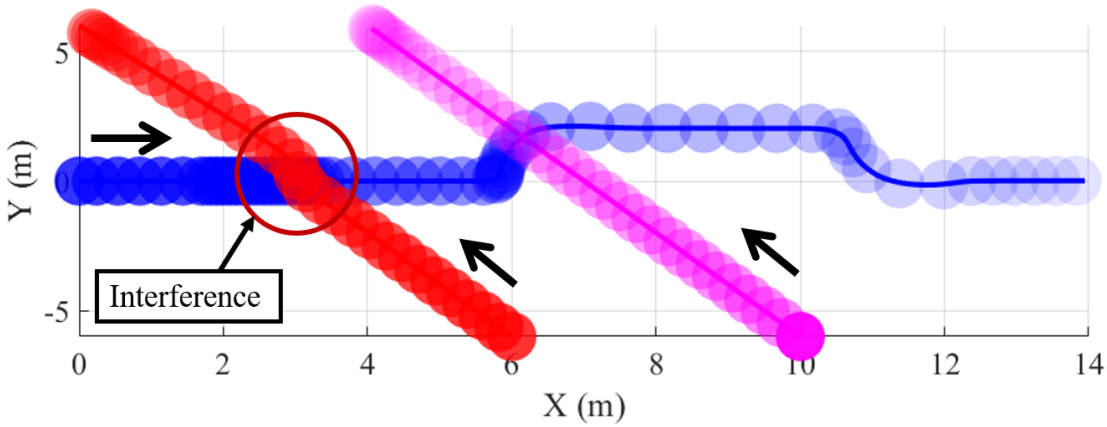


(d) Distance to human.

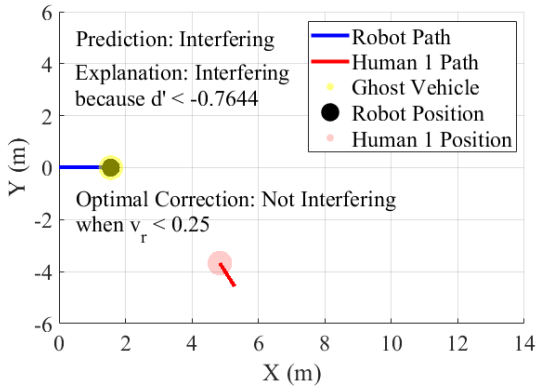
Figure 3.10: Baseline simulation.

In Fig. 3.11, we show the effects of online validation and learning by performing a simulation with DTs trained on an incomplete training dataset. The robot's goal is $(14, 0)$m and the humans in this simulation take identical paths in succession to test whether the robot has improved its behavior. The robot makes an incorrect decision at first, applying the explanation and correction shown in Fig. 3.11(b), only slowing down to $v_r < 0.25$m/s. Both test cases (Sec. 3.3.6) are violated, shown by the deviation in the red path of Fig. 3.11(a) and the observation (red point) outside the partial convex hull in Fig. 3.11(d). Note that partial 3-dimensional convex hulls are shown for visualization purposes, due to the high dimensionality of our attributes, $\boldsymbol{\alpha} \in \mathbb{R}^8$. The magenta path in Fig. 3.11(a) shows no interference, as a different correction was selected (Fig. 3.11(c)), since the observations are now included in the local data (Fig. 3.11(e)).

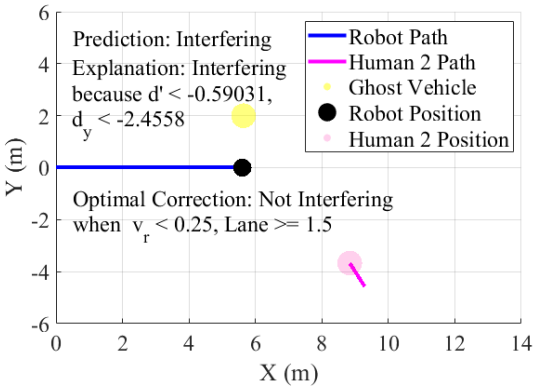
We also extensively test our approach in handling multiple people at a time, shown in Fig. 3.12. The robot navigates through 10 people, changing its lane (ℓ) and velocity (v_r) to avoid interfering. This test was run for 50 trials with random human trajectories. All decision tree operations in these simulations took between $30 - 90$ms. Our approach successfully eliminated interferences 72% of



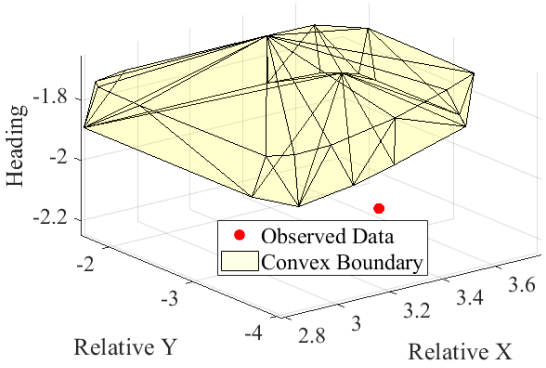
(a) Paths of humans (red, magenta) and robot (blue).



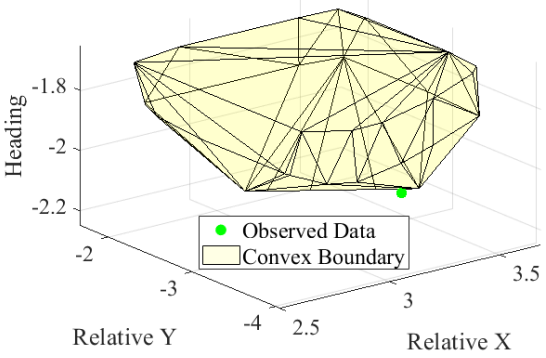
(b) Incorrect DT operations.



(c) Corrected DT operations.



(d) Partial convex hull prior to updates.



(e) Partial convex hull after updates.

Figure 3.11: Simulation of runtime validation and updating.

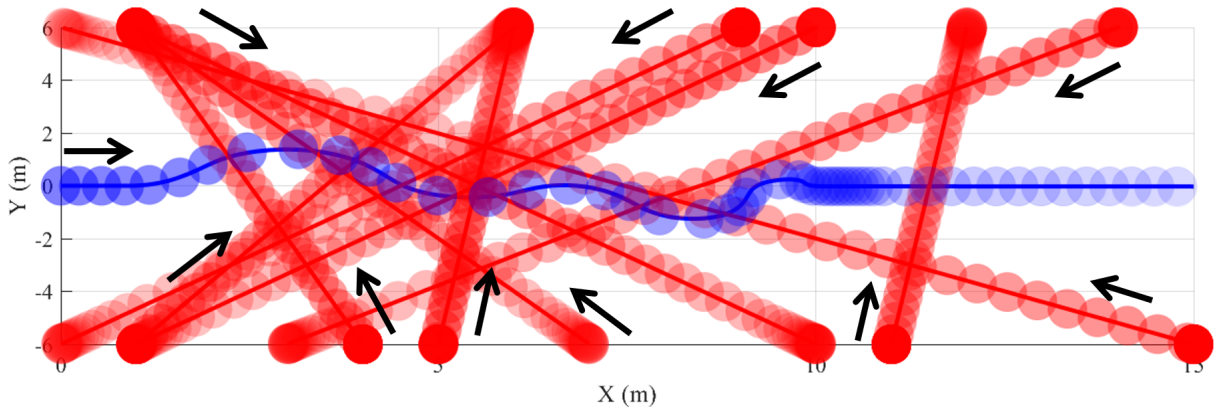


Figure 3.12: Human (red) and robot (blue) paths in a multi-actor simulation

the time, with an average minimum distance of $\delta_{\min} = 1.58$m. Where interference occurred, we observed that the robot was in dense crowds, negotiating with on average $N_h(t) \geq 7$.

3.5 Experiments

The proposed approach was also validated experimentally on a Clearpath Robotics Ridgeback Omnidirectional Platform (see Fig. 2.1) in indoor environments. In the first experiments, a VICON motion capture (MOCAP) system was used to obtain robot and human states. In the second experiment, the robot uses an on-board ASUS Xtion RGB-D camera with the SPENCER people tracking package [63].

3.5.1 MOCAP Experiments

In the first experiment shown in Fig. 3.13, the robot predicts, explains, and takes corrective actions proactively to avoid the human. The robot moves from $(-2.5, 0)$m to $(2.5, 0)$m at a nominal velocity of $v_r = 0.6$m/s, and the distance threshold $\delta_{th} = 1$m. The robot predicts $\mathcal{P}_h(t) = \lambda$ and explains:

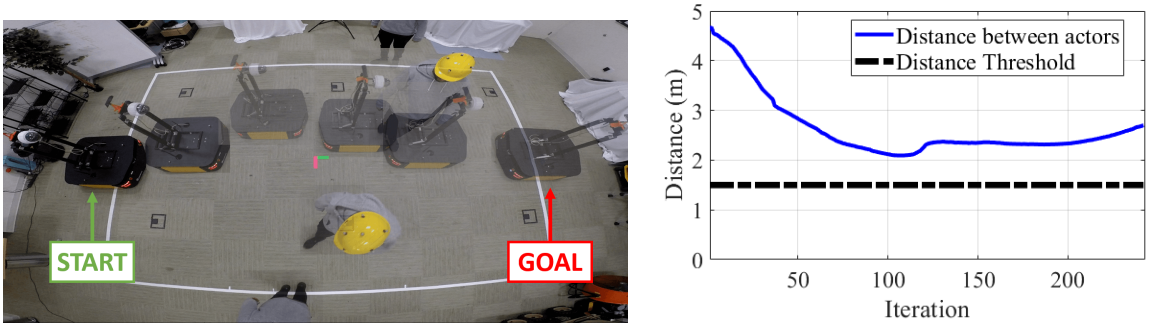
$\mathcal{E}_h(t) =$ ***Interfering*** because: $\{d' < 0.21, d_x > 2.59\}$

With the correction tree, the robot computes:

$\boldsymbol{\mathcal{C}}_h(t) =$ ***Not Interfering*** when: $\{v_r \geq 0.5, \ell > 0.5\}$

Thus, the corrective action is to maintain nominal velocity, and move to the lane in the positive y-direction, to avoid interfering at the center of the environment. The minimum distance between actors is $\delta_{\min} = 2.09$m (Fig. 3.13(b)) , never nearing the distance threshold.

In the experiment shown in Fig. 3.14, we examine the effects of online validation and updating.



(a) Snapshots of experiment.

(b) Distance between actors.

Figure 3.13: Snapshots and distances of lab experiment.

As a proof of concept, we removed a large portion of our training data, and as shown in the first run in Fig. 3.14(c) we observed that the robot moved incorrectly toward the human. The distance threshold is violated, $\delta_{\min} = 0.91$m (Fig. 3.14(e)), and the person alters his path, reacting to the robot's interfering behavior. After the model is updated and reinforced at runtime, the robot makes the appropriate correction and no interference is observed, with $\delta_{\min} = 2.13$m (Fig. 3.14(c-d)).

In Fig. 3.15, we show the effectiveness of our approach with two people in the lab environment. The robot has the goal to reach $(2.5, 0)$m and successfully avoids interference with both people at once, even in a small space, showing that our approach scales to accommodating multiple actors at the same time. The trajectories for all agents are shown in Fig. 3.15(e), and the minimum distance between any human and robot was 1.23m, which is above the threshold $\delta_{th} = 1$m.

3.5.2 On-Board Sensing Experiment

To demonstrate the applicability of our approach outside MOCAP settings, we deployed our technique on the same robot using only the on-board RGB-D and Lidar sensors to identify and track surrounding humans and localize itself. Fig. 3.16 shows the results for this experiment in which the robot is able to successfully avoid interference with surrounding people and reaches its goal without violating the distance threshold despite noisy camera measurements and uncertain person detection and tracking.

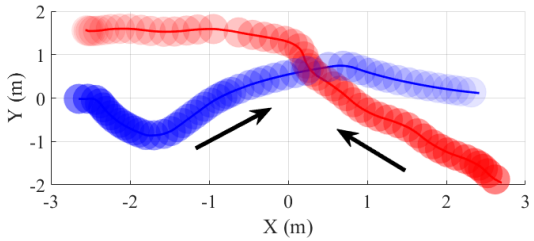
3.6 Discussion

In this work, we have presented a novel approach for interpretable prediction and planning of a robot in a co-robotic environment. We relax the requirement of explicitly predicting human paths, and instead directly predict, explain, and find counterfactual rules for interfering behaviors with binary decision trees and a library of pre-trained trajectories. Unique from other works in this field, we validate robot behaviors to update the predictive model at runtime, resulting in improved behaviors in future operations. While we focused on human-robot operations in this work, our framework works for any motion planning operation with multiple actors. The results overall show desirable and explainable robot behaviors among dynamic actors behaving in different ways.

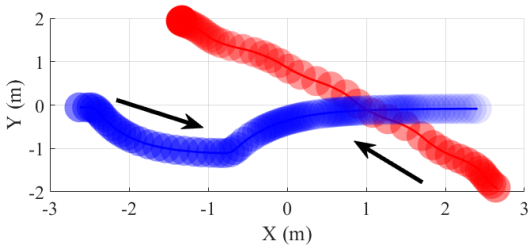
3.6.1 Limitations and Future Directions

While we found that our approach shows good results, we note that performance can decrease in very dense ($N > 7$) crowds, and this is primarily because the binary classification approach is, by design, conservative and inflexible in differentiating between excessive interference and slight interference beyond the threshold set in the training. Furthermore, since predictions are not made during transition states, the appearance of new actors in that time may result in a robot that switches into the fail safe mode too much. But it should be noted that causing some interference is not necessarily unnatural or socially unacceptable, so this can be mitigated further by relaxing the requirement about interfering with nearby actors and providing explanations and corrections that are flexible to different desired priorities of the robot. Finally, we note that the method currently used requires searching through a dataset to update and learn new behaviors online. This, however, can get computationally expensive without limiting the size of the dataset. At the same time, limiting the dataset may remove useful information the robot could use to better inform planning.

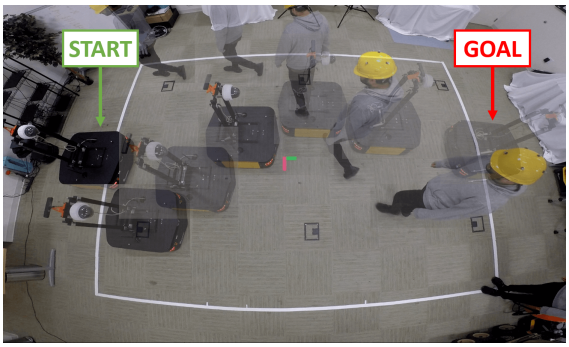
To handle these challenges, we leverage the idea of training the DT monitor with probabilities measured, collected, and updated at runtime to assist decision-making in traditional planners and controllers. We also modify the definition of interference to better prioritize desired robot behaviors in multi-actor environments. These methods are introduced in the next chapter.



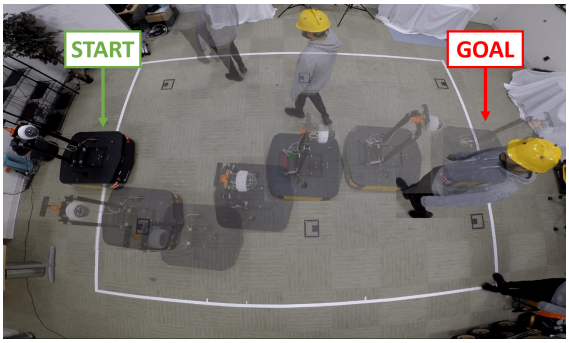
(a) Experiment paths of human (red) and robot (blue) with an incomplete subset of training data.



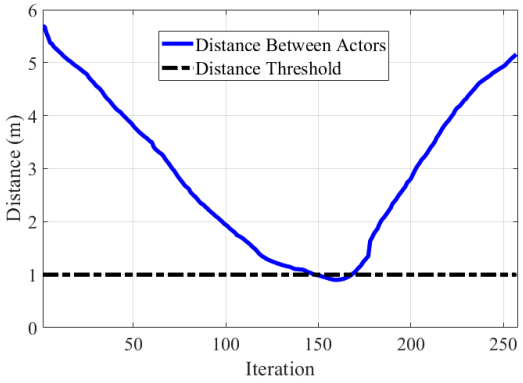
(b) Experiment path of humans (red) and robot (blue) after including the previous observation at runtime.



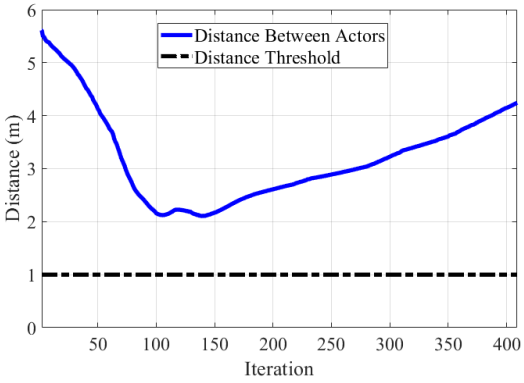
(c) Experiment snapshots before updates.



(d) Experiment snapshots after updates.

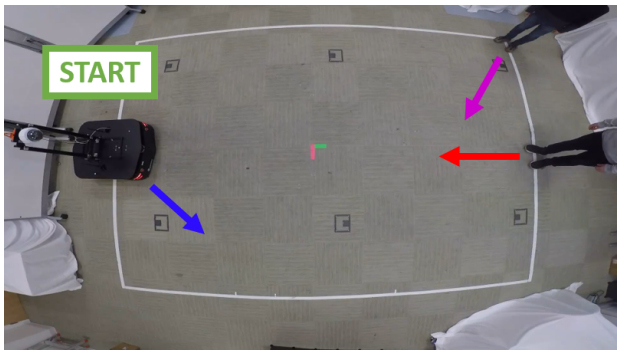


(e) Distance between actors before update.

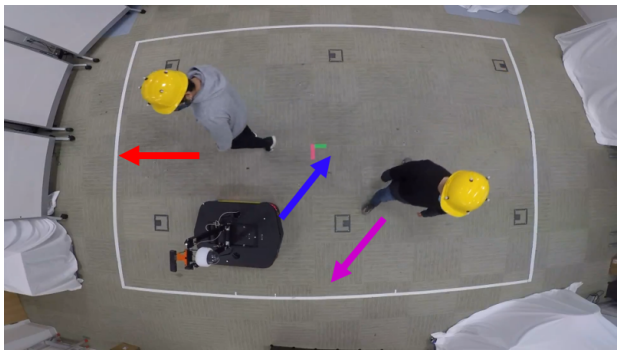


(f) Distance between actors after update.

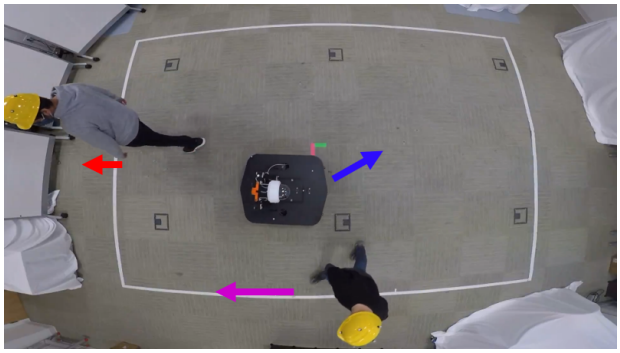
Figure 3.14: Experiment showing the effects runtime updates.



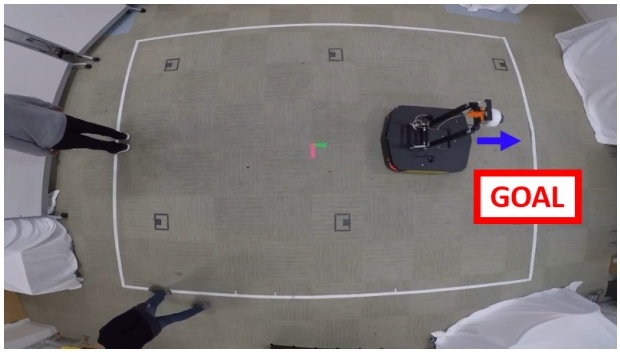
(a) Initial positions of actors.



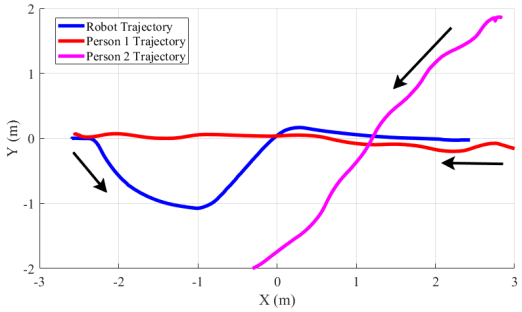
(b) Robot performs corrective behavior.



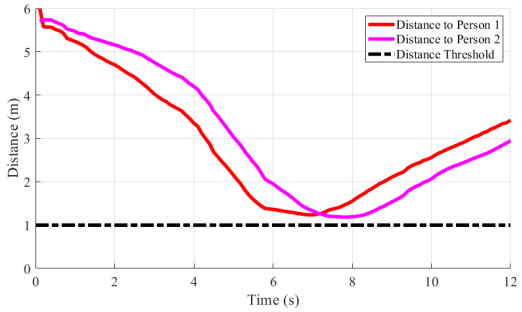
(c) Intermediate positions of actors



(d) Final positions of actors.

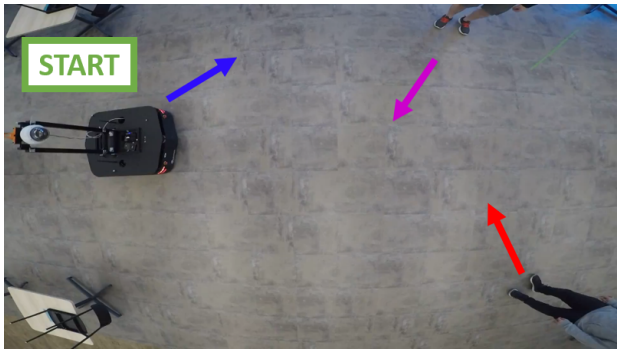


(e) Trajectories of actors.

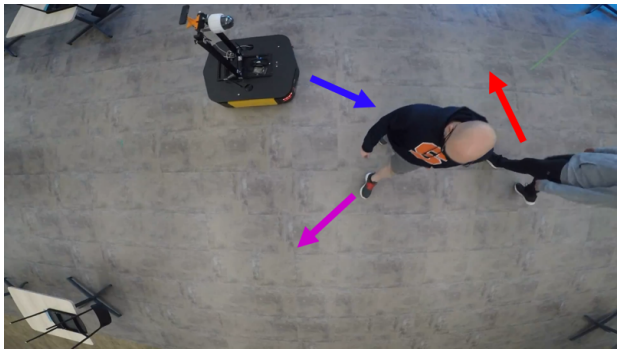


(f) Distance between actors.

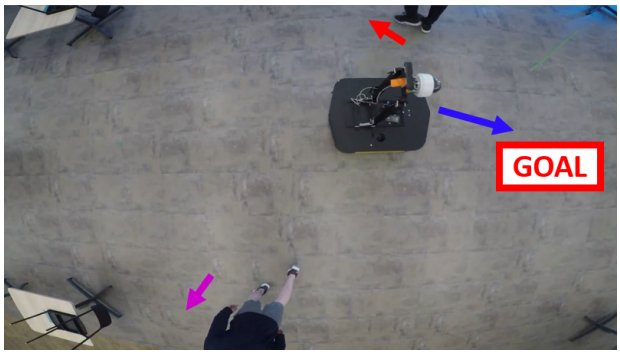
Figure 3.15: Snapshots, trajectories and distances of 2-person lab experiment.



(a) Initial positions of actors.

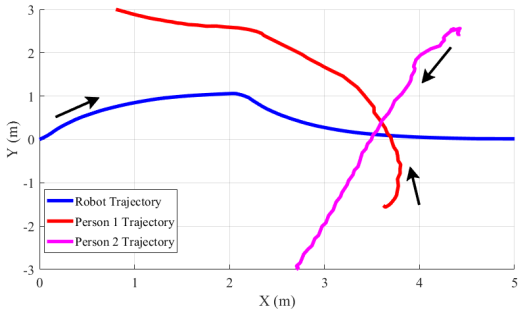


(b) Robot performs corrective behavior.

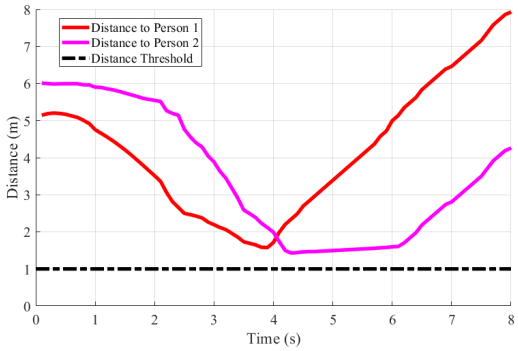


(c) Actors arrive at final positions

(d) Robot first person view.



(e) Trajectories of actors.



(f) Distance between actors.

Figure 3.16: Snapshots and trajectories of 2-person lab experiment.

Chapter 4

Interpretable Adaptation of Virtual Physics-based Planner for Social Navigation

In this chapter, we expand on the decision tree-based (DT) interpretable monitoring framework presented in the previous chapter for priority-aware adaptation of a baseline virtual physics-based (VP) planner. We note that these planners are efficient and scale well to avoiding collisions with multiple agents, but leave the robot susceptible to getting stuck in local minima. In dynamic environments, such issues can cause interference with nearby actors, an thus, we leverage our interpretable monitor here to adapt the parameters of such planners to ensure non-interfering behaviors. One key addition to our approach includes a Hidden Markov Model-inspired (HMM) mechanism to store new data collected at runtime to update and improve predictions and planning faster than the data collection methods introduced in the previous chapters. We also extend the counterfactual analysis to adapt corrective actions to the robot's priorities, which can vary in different environments with different applications. These desired priority behaviors are taken into account when performing counterfactual analysis within our framework. We show that the interpretable monitoring framework, in addition to serving as the high-level motion planner as in the previous chapter, can be flexible enough to adapt and adjust for failures of robotic planning systems. Our framework is validated with multiple simulations and experiments consisting of a UGV and a UAV equipped with the VP planner that successfully navigates without interfering with the paths of multiple dynamic actors. This work has been published in the Robotics and Automation Letters (RA-L).

4.1 Introduction

As humans, when navigating in the presence of others, we combine competing notions of accommodating each other's paths while also maintaining our own path or desired speed. We consider these factors simultaneously to navigate in natural, socially-acceptable ways. Moreover, we prioritize how we accommodate others based on our specific environment or task. For example, a doctor rushing to aid an injured person will prioritize speed as much as safely possible. Importantly, we make these decisions without explicitly predicting the paths of other actors, and we can explain and understand why we make these decisions, continually learning and adapting our behaviors as we experience different scenarios. If a robot could reason about its behavior and plan corrective actions in a similar way, it could generate similarly natural and intuitive motion around other actors, while maintaining priorities related to how much it should accommodate to surrounding actors.

In any cases, mobile robots that share environments with humans are typically already equipped with a basic reactive motion planner that is used to reach goals in a safe manner, usually without colliding with perceived obstacles in the environment. With these basic planners, however, motion planning may result in undesirable deviations and potentially, collisions.

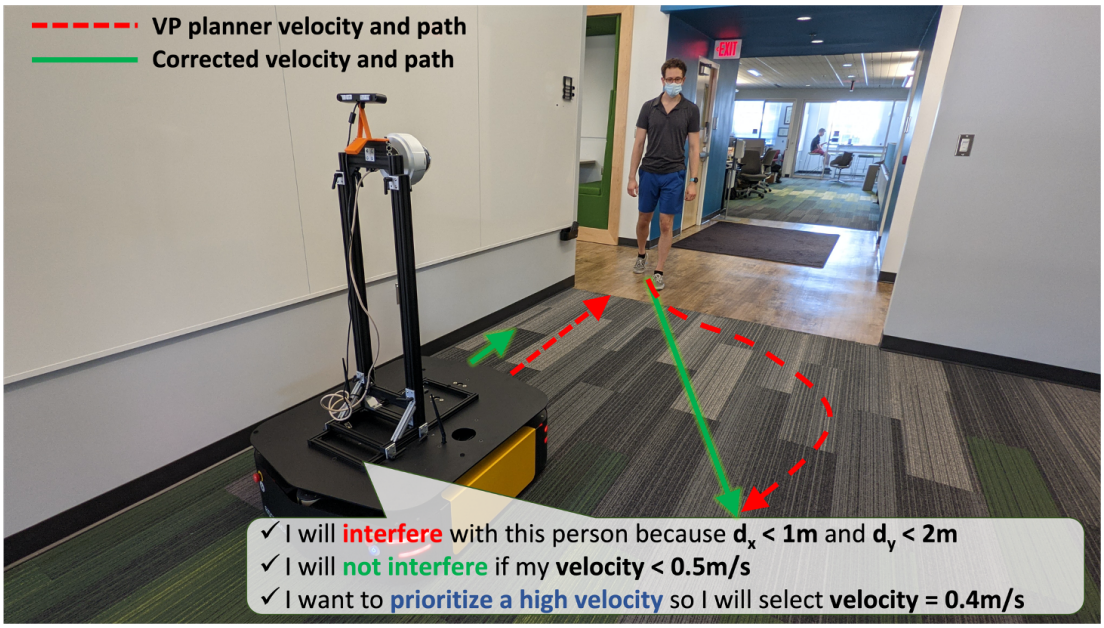


Figure 4.1: In our proposed approach, a robot predicts, explains and finds a priority-aware corrective action on top of a virtual-physics planner to avoid interfering with oncoming actors

Virtual physics-based (VP) planners, including artificial potential fields (APF), are well-known examples of this type of planner because they are easy to implement and are very efficient, typically scaling well to handling multiple obstacles, irrespective of whether these obstacles are static or dynamic. Robots using these reactive planners, however, can get stuck in local minima or in prolonged divergence from the goal, as a result of not accounting for dynamism of the obstacles. On the other hand, if this type of planner can be adapted in a predictive and proactive way, it is possible to generate desirable accommodating behaviors. Then, the objective of our framework becomes predicting whether a failure case will arise, understanding why such a failure will arise, and finding a method to proactively correct the behavior of the system. This correction, in addition, can be fine tuned to the priorities of the robot, which can include maintaining a target velocity or maximizing distance from surrounding actors. Finally, another consideration in this work is that our previous methods to update the model at runtime have relied on collecting and storing a large window of data. In this work, we relax this requirement by leveraging principles from the HMM presented in Chapter 2. Instead of storing observed data directly, we build and update at runtime a probability matrix based on our observation. This probability matrix is used directly to inform local DT training. In Fig. 2.1, we show a motivating example for our work, in which a robot equipped with our approach predicts, explains, and finds a priority-based correction to avoid interfering with an oncoming person.

This work presents three main contributions: 1) the design of an interpretable DT-based monitor that predicts, explains, and finds corrections when a robot will interfere with nearby dynamic actors; 2) the formulation of a priority-based reward function to identify the correction that optimizes how the robot should behave; 3) an HMM-based model to build and update decision trees at runtime and improve predictions and planning over time without storing a large amount of data.

4.2 Preliminaries

Let us consider a mobile robot equipped with a VP planner that is tasked to navigate to a goal in the presence of dynamic actors. Specifically, our planner is inspired by an efficient and scalable virtual spring-mass-damper system borrowed from our previous work [78], in which the input $\boldsymbol{u}$ is comprised of an attractive force that draws the robot towards its goal and repulsive forces that send

the robot away from the obstacles. Generally, this formulation is in terms of forces and accelerations, but can be cast as kinematic constraints, depending on the capabilities and inputs of the robots, many of which typically only accept velocity commands. Thus, for ease of implementation on a wide range of robots, we cast the input $\boldsymbol{u}$ as a velocity which is governed by attractive and repulsive effects.

The attractive input that moves the robot from its position $\boldsymbol{p}_r(t)$ towards the goal $\boldsymbol{p}_g$ is computed as follows:

$$\boldsymbol{u}_{att}(t) = k_{att}(||\boldsymbol{p}_r(t) - \boldsymbol{p}_g||)\vec{\boldsymbol{d}}_{\boldsymbol{g}} \tag{4.1}$$

where $\vec{\boldsymbol{d}}_g$ is the unit vector directed towards the goal and k_{att} is the attractive spring constant. In this work, the resulting velocity vector is limited by a maximum target speed: $||\boldsymbol{u}_{att}(t)|| \leq v^*$. We assume here that the robot is able to quickly reach its target speed, and the desired effect is that the robot slows down and stops once it is close to the goal.

The repulsive inputs are computed as follows:

$$\boldsymbol{u}_{rep}(t) = \begin{cases} k_{rep}(d_{ho}(t))\vec{\boldsymbol{d}}_{\boldsymbol{ho}}, & d_h(t) \leq l_o \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where k_{rep} is a repulsive spring constant, $d_{ho}(t) = l_h(t) - l_o$ and $l_h(t) = ||\boldsymbol{p}_r(t) - \boldsymbol{p}_h(t)||$ is the distance between robot and dynamic actor h, l_o is the reaction distance threshold, and $\vec{\boldsymbol{d}}_{\boldsymbol{ho}}$ is a unit vector in the direction away from the actor.

We then combine the attractive and repulsive effects along with a damping term with coefficient c_d to compute the input:

$$\boldsymbol{u}(t) = \boldsymbol{u}_{att}(t) + \sum_{j=1}^{n} \boldsymbol{u}_{rep}(t) - c_d\boldsymbol{u}(t-1) \tag{4.3}$$

VP planners, as it is well-known, are a reactive approach for motion planning and robots using these types of planners can often get stuck in local minima or experience prolonged divergence from the goal. Local minima occur when repulsive and attractive inputs are equal leading to $\boldsymbol{u} = 0$, and prolonged divergence occurs when a repulsive input sends the robot away from the goal and is not

approaching $\boldsymbol{u}_{rep} = 0$ quickly enough for the robot to converge to its target. Shown in Fig. 4.2 are examples of two trajectories in which the VP planner results in a failure (Fig. 4.2(a-b)) due to a local minima and prolonged divergence from path, and one example of a successful VP motion plan (Fig. 4.2(c)). Figs. 4.2(d-f) display the magnitude of the repulsive input on the robot for each of these trajectories.

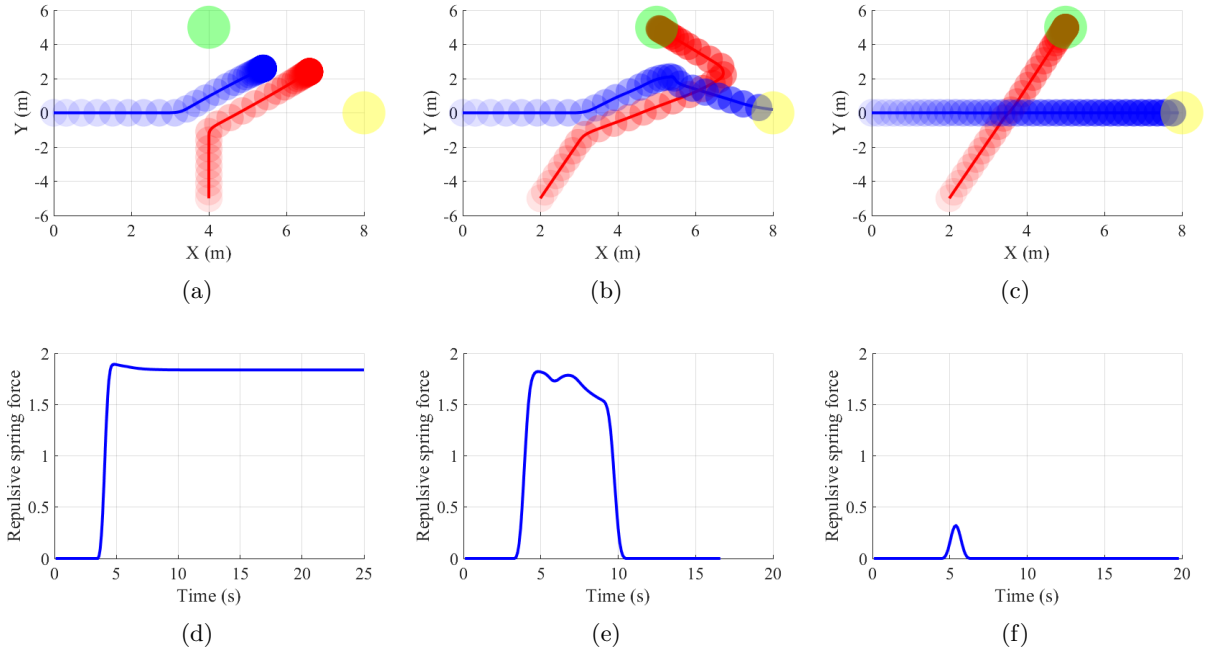


Figure 4.2: Examples of VP planner trajectories and repulsive inputs. The trajectories are shown by blue (robot) and red (actor) markers that fade as time passes. Yellow (robot) and green (actor) markers represent the goals.

4.3 Problem Formulation

Consider a mobile robot equipped with the aforementioned VP planner navigating a shared environment with other dynamic actors. With the premises presented in Sec. 4.2, we would like to directly predict when the VP planner will lead to interference and plan motion accordingly to correct the behaviors of the robot. Formally, the problem is:

Problem 4.1 (Interpretable Interference Monitoring) *Design a policy to predict and explain future interfering interactions between a robot and nearby actors and find corrective control actions,*

$\boldsymbol{u}_{corr}$, *for a VP planner such that the following non-interfering conditions are met:*

$$\begin{aligned} ||\boldsymbol{p}_r(t) - \boldsymbol{p}_i(t)|| &> \delta_s, \ \forall i = 1, \ldots, N_h(t) \\ ||\boldsymbol{p}_i(t) - \boldsymbol{p}_i^*|| &\leq \sigma_h, \ \forall i = 1, \ldots, N_h(t) \\ ||\boldsymbol{p}_r(t) - \boldsymbol{p}_r^*|| &\leq \sigma_r \end{aligned} \tag{4.4}$$

where $\boldsymbol{p}_r(t)$ *and* $\boldsymbol{p}_i(t)$*, represent the robot and actor* i *positions,* $\boldsymbol{p}_r^*$ *and* $\boldsymbol{p}_i^*$ *are the reference positions of robot and actor* i *along their respective desired paths, and* δ_s *is a minimum distance threshold that can be obtained through testing (in this work it is set to* $1m$*), and* σ_r *and* σ_h *are deviation thresholds for robot and actor paths, respectively. In this work, we assume the desired path of actors can be estimated by analyzing changes in direction of motion and checking the minimum distance between robot and actor within the robot's sensing range.*

The robot also has different behaviors depending on the assigned priorities. For example, in a conservative case, a robot may accommodate actors more, while a more aggressive robot may prioritize moving faster, accommodating less to the motion of nearby actors. In this work, for ease, we assume that the robot considers commonly observed behavioral priorities [57, 45] that pertain to distance, deviation, and velocity, as follows:

1. Maintain a safe distance from dynamic actors (R_δ)
2. Minimize deviation from the robot's desired path (R_{σ_r})
3. Minimize actor deviation caused by the robot (R_{σ_h})
4. Minimize deviation from robot's target speed (R_v)

Given the above priority considerations, we can then formulate the following reward function:

$$R(\boldsymbol{\beta}) = \beta_1 R_\delta + \beta_2 R_{\sigma_r} + \beta_3 R_{\sigma_h} + \beta_4 R_v, \tag{4.5}$$

where $\boldsymbol{\beta} = [\beta_1 \ \beta_2 \ \beta_3 \ \beta_4]$ is the set of parameters that defines the weights given to each of the 4 considerations. It should be noted that different priorities can be considered beyond those mentioned above.

Then, the problem becomes the following:

Problem 4.2 (Priority-based Motion Planning) *Find a corrective input action within* $\boldsymbol{u}_{corr}$ *from Problem 1 for the robot that maximizes the objective reward function given a particular set of priorities* $\boldsymbol{\beta}$*:*

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u}} R(\boldsymbol{\beta}) \tag{4.6}$$

Finally, a corollary problem we investigate is how the robot can learn and improve its behavior at runtime in a manner that does not become more computationally expensive over time, and is viable for long term deployment.

4.4 Approach

Our proposed priority-based interpretable monitoring and planning framework follows the architecture in

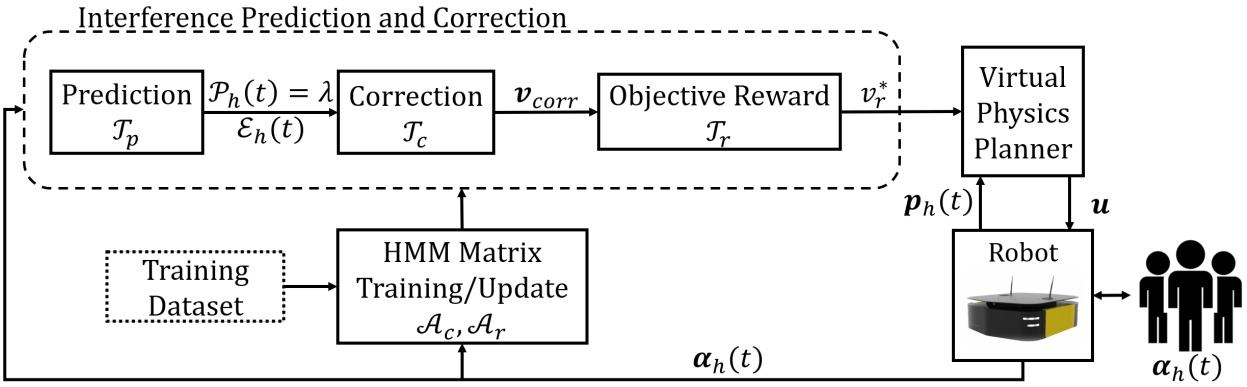


Figure 4.3: Block diagram of our priority-based interpretable monitoring and planning framework.

At the core of our framework, a series of decision trees is used to: 1) predict and explain interference between the robot and the actors, 2) obtain corrections and 3) select the proper correction for the virtual physics planner based on different priority metrics imposed on the robot. To enable runtime learning and adaptation, a Hidden Markov Model (HMM) is trained both offline and at runtime to characterize probabilities of interference. In the event that the DT predicts that the VP planner will not interfere, then no correction is needed and the robot plans motion according to (4.3). In what follows we will discuss in more detail each element of Fig. 4.3.

4.4.1 Probability-based Decision Tree Theory

Decision trees (DTs) are a form of supervised learning that are interpretable, due to their white-box models [16]. DTs take in a set of input variables (attributes) to predict a target output. When the output takes a discrete set of values (classes), the DT is a classification tree. In these trees, the outermost nodes correspond to class labels and internal nodes represent conjunctions of features that return the respective class labels. DTs where the output takes continuous values are known as regression trees [37], and the outermost nodes in these trees correspond to real numbers. In this work, classification trees are used to predict and explain a future interference, and regression trees are used to estimate priority rewards to decide how to correct robot behaviors.

Typically, DTs are trained with a dataset of historical observations containing input variables and associated labels. A key feature in this work is that DTs are both constructed and constantly updated at runtime, meaning that new data are constantly observed and incorporated into the construction of DTs. Storing historical observation data can become computationally expensive over time, bringing about the need for a faster approach to update and build DTs at runtime.

To this end, we propose a probability model for fast updating and construction of DTs. We take inspiration from the emission matrices within Hidden Markov Models (HMM) [81], in which the state space of the system, $\mathcal{S} \in \mathbb{R}^N$, includes a finite set of unique states $s_i \in \mathcal{S}$, $i = 1, \ldots, N$, and each state can be associated to an emission, $\mathcal{G} \in \mathbb{R}^M$ and $g_k \in \mathcal{G}$, where $k = 1, \ldots, M$, with $M \in \mathbb{N}$. Given N unique states and M unique emissions, the right-stochastic matrix $\mathcal{A} \in \mathbb{R}^{N \times M}$, in which the sum of rows is 1, lists the probability a_{ik} of obtaining an emission g_k given a state s_i, and is formed as follows

$$\mathcal{A} = \begin{bmatrix} a_{11} & \ldots & a_{1M} \\ \vdots & \ddots & \\ a_{N1} & & a_{NM} \end{bmatrix} \tag{4.7}$$

In a typical HMM framework, values within $\mathcal{A}$ do not account for the fact that there may be very few occurrences of a particular state, assigning very high probabilities with very little information, which may lead to inaccurate classification. We instead include this consideration by treating emissions as classes and computing classification probabilities that connect a particular state s_i to a class g_k by leveraging a generalized logistic function (GLF) [23], which is an extension of the sigmoid function,

of the form:

$$a_{ik}(x) = \frac{L}{1 + Qe^{-b(x-x_0)}}, \tag{4.8}$$

where L is the maximum value; since this function is used to compute and update probabilities between 0 and 1, a natural choice for this value is $L = 1$. The logistic growth rate is defined by b and x_0 is the midpoint of the curve. The parameter Q can be treated as an initial bias on the midpoint ($x = 0$) of the function. If $Q > 1$, the GLF has a bias towards a lower probability, $a_{ik}(0) < 0.5$, and vice versa. When $Q = 1$, there is no bias and $a_{ik}(0) = 0.5$. The value of Q is a user defined parameter that depends on the application and can be used to tune how conservatively a classification model makes decisions.

The value of x must capture the desired comparison between output classes. In this work, x is computed as a weighted difference between observations of a certain state in each class:

$$x = \begin{cases} \frac{N_{s_{ik^*}} - N_{s_{ik}}}{N_{s_i}} \cdot \min(1, \frac{N_{s_i}}{N_l}) & N_{s_i} > 0 \\ 0 & N_{s_i} = 0 \end{cases} \tag{4.9}$$

where $k^* = \neg\lambda$ and $k = \lambda$ represent the classification outputs in our work, $N_{s_{ik^*}}$ and $N_{s_{ik}}$ are the number of non-interfering and interfering observations at state s_i, respectively, and N_{s_i} is the total number of observations of s_i. The value N_l is a buffer chosen to reduce the impact in cases with very few observations while allowing cases with sufficient observations ($N_{s_i} \geq N_l$) retain their intrinsic values. Shown in Fig. 4.4 is the GLF that we obtained via experimental testing for this work, where

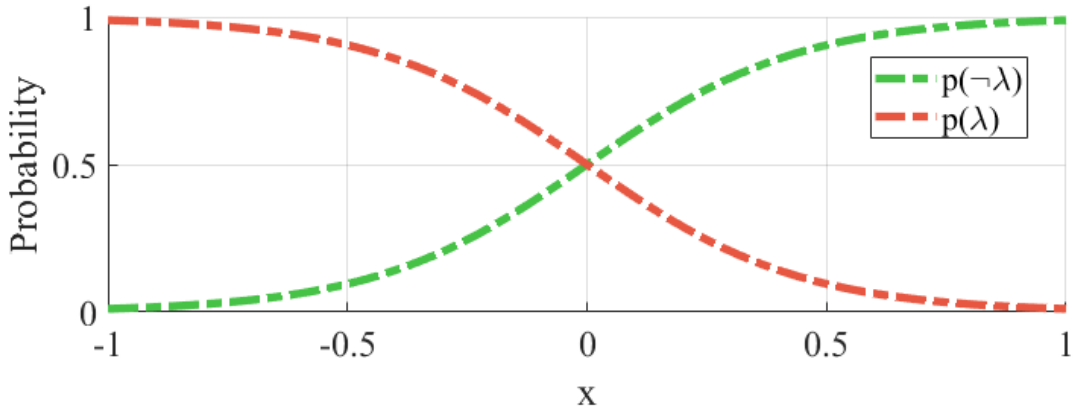


Figure 4.4: Generalized logistic function used to compute $P(\neg\lambda)$. Also shown is the complementary curve for $P(\lambda)$

the midpoint is $x_0 = 0$, the parameter $Q = 1$, and the growth rate is $b = 4.5$, giving the following expression: $p(x) = \frac{1}{1+e^{-4.5x}}$.

Since growing DTs requires a dataset containing input states and outputs, we leverage $\mathcal{A}$ to generate a fixed number of data, in which the distribution of outputs corresponds to class probabilities in $\mathcal{A}$. An important benefit of using this method to build and update DTs and probabilities at runtime is that the only data storage requirements for each state are the values contained within (4.9), which can be adjusted at runtime by updating the number of state and output observations, and recalculating probabilities with (4.8) and (4.9). The complexity of this method does not increase with time, and is viable for long term deployment.

4.4.2 HMM and DT Training

The training data used to build HMM matrices consists of trajectories of a robot using the aforementioned VP planner with different target speeds in the presence of one actor moving from various initial positions to various goals. In training and simulations, actors are also controlled by the presented VP planner, to account for the fact that other actors are non-hostile and will alter their paths to avoid collisions with the robot.

The attributes (inputs to DTs) are based on the available sensor data about the robot and actors. Specifically, we leverage the following attributes that we have experimentally validated in our previous work [80]:

$$\boldsymbol{\alpha} = [\ d_x \quad d_y \quad \theta \quad v_h \quad v_r\ ] \tag{4.10}$$

where d_x, d_y, and θ, are relative x-y positions and heading and v_h and v_r are the actor and robot speeds, respectively. Each unique set of attributes represents one of the states, or rows, within the probability matrix: $\boldsymbol{\alpha}_i = s_i \in \mathcal{S}$, with $i = 1, \ldots, N$. The collected attributes are limited to a small sensing range around the robot since the goal of the robot is to avoid interfering with nearby actors. The attributes are discretized uniformly to prevent an infinite or exploding state space. While uniform discretization is leveraged given the smaller sensing range, it is worth noting that for a robot considering a much larger sensing range or a different application, a variable discretization may be more desirable, so that predictions regarding actors closer to the robot are more accurate and granular than those further from the robot.

The output classes are either interfering (λ) or not interfering ($\neg\lambda$), and are assigned based on a violation of any condition in (4.4) at any point in a trajectory. The regression outputs describe how well a set of attributes performs for a certain set of priorities (Prob. 2 in this work). Below, we define how each of the rewards within (4.5) are computed.

The safe distance reward, R_δ, is constructed such that the maximum reward of 1 is assigned when the threshold constraint δ_s is not violated and the reward proportionally decreases as a violation worsens:

$$R_\delta = \begin{cases} \frac{||\boldsymbol{p}_r(t) - \boldsymbol{p}_h(t)||}{\delta_s}, & ||\boldsymbol{p}_r(t) - \boldsymbol{p}_h(t)|| < \delta_s \\ 1, & ||\boldsymbol{p}_r(t) - \boldsymbol{p}_h(t)|| \geq \delta_s \end{cases} \tag{4.11}$$

The reward for actor path deviation is assigned differently as follows, since the desired effect is lower deviation:

$$R_{\sigma_h} = \begin{cases} 0, & ||\boldsymbol{p}_h(t) - \boldsymbol{p}_h^*|| > \sigma_h \\ e^{-\frac{||\boldsymbol{p}_h(t) - \boldsymbol{p}_h^*||}{\sigma_h}}, & ||\boldsymbol{p}_h(t) - \boldsymbol{p}_h^*|| \leq \sigma_h \end{cases} \tag{4.12}$$

where σ_h represents the maximum permitted deviation that satisfies conditions in (4.4). The robot path deviation reward R_{σ_r}, is similar to the actor deviation reward, but instead uses robot current and desired position.

The velocity reward can be represented by any function that has a global maximum at the desired speed and decreases as the robot's speed diverges from v_{des}. We capture this effect with a quadratic function that has a vertex at $(v_{des}, 1)$, as follows:

$$R_v = -a(v_r - v_{des})^2 + 1 \tag{4.13}$$

where a is a constant that determines the rate at which the reward decreases.

In the case where any one of the individual rewards is more important, that term should be weighted higher. For example, the objective for an actor-focused case, in which the robot prefers yielding to actors and incurring more deviation in its own path and velocity, can have the following set of parameters, $\boldsymbol{\beta} = [0.2\ 0.1\ 0.5\ 0.2]$, further penalizing states that are predicted to cause more actor deviation. $\beta_i = 0.25\ \forall\ i$.

During training, an observed set of attributes is rounded to the nearest state, $s_i \in \mathcal{S}$, which is

then associated to a class label $(\lambda, \neg\lambda)$ and a priority based reward output (4.5). Then, the class probabilities are computed with (4.8) and (4.9) to populate a classification matrix $\mathcal{A}_c$, and the regression matrix, $\mathcal{A}_r$ is built based on the priority-based rewards obtained during training.

4.4.3 Prediction and Explanation

At runtime, to predict and explain interference with a surrounding actor, we build local decision trees with an actor's observed attributes $\boldsymbol{\alpha}_h(t)$. Local trees are more desirable for prediction and explanation, because DTs constructed using the entire state space can contain irrelevant information, reducing the quality of predictions and explanations [37]. The local tree, $\mathcal{T}_p^h$, is trained using a subset of states $\boldsymbol{s}_h(t)$ from $\mathcal{S}$ that are within a distance Δ of the attributes of $\boldsymbol{\alpha}_h(t)$:

$$\boldsymbol{s}_h(t) \subset \mathcal{S} \text{ s.t. } ||\boldsymbol{\alpha}_h(t) - s_i|| \leq \Delta \ \forall s_i \in \mathcal{S} \tag{4.14}$$

The parameter Δ defines how close the local training states are to the observed state, and this value should be chosen based on the quality of the training set. It should be noted that too low a Δ may remove relevant data, while too large a Δ provides little benefit over a global tree. Then, the prediction $\mathcal{P}_h(t) \in [\lambda, \neg\lambda]$ is obtained as follows: $\mathcal{P}_h(t) = \mathcal{T}_p^h(\boldsymbol{\alpha}_h(t))$.

An explanation $\mathcal{E}_h(t)$ is then computed by traversing the path Γ from the root of the tree, $\mathcal{V}_0$, to a prediction leaf, $\mathcal{V}_p$, taking the conjunction of each split criterion, c, for the N_i nodes along the path:

$$\mathcal{E}_h(t) = \bigwedge_{k=1}^{N_i} c_k \quad \text{with} \quad \Gamma \mid \mathcal{P}_h(t) \tag{4.15}$$

Traversing all paths, $\Gamma_j \in \boldsymbol{q}$ with $j = 1, \ldots, N_{\boldsymbol{q}}$, that lead to the opposite decision provides a set of counterfactuals, $\boldsymbol{\mathcal{C}}_h(t)$:

$$\boldsymbol{\mathcal{C}}_h(t) = \bigvee_{j=1}^{N_{\boldsymbol{q}}} \bigwedge_{k=1}^{N_j} c_k \quad \text{with} \quad \Gamma_j \mid \neg\mathcal{P}_h(t) \tag{4.16}$$

where each path Γ_j contains N_j nodes to the leaf [80].

In Fig. 4.5 we show an example of a decision tree that predicts and explains a future interference with one approaching actor.

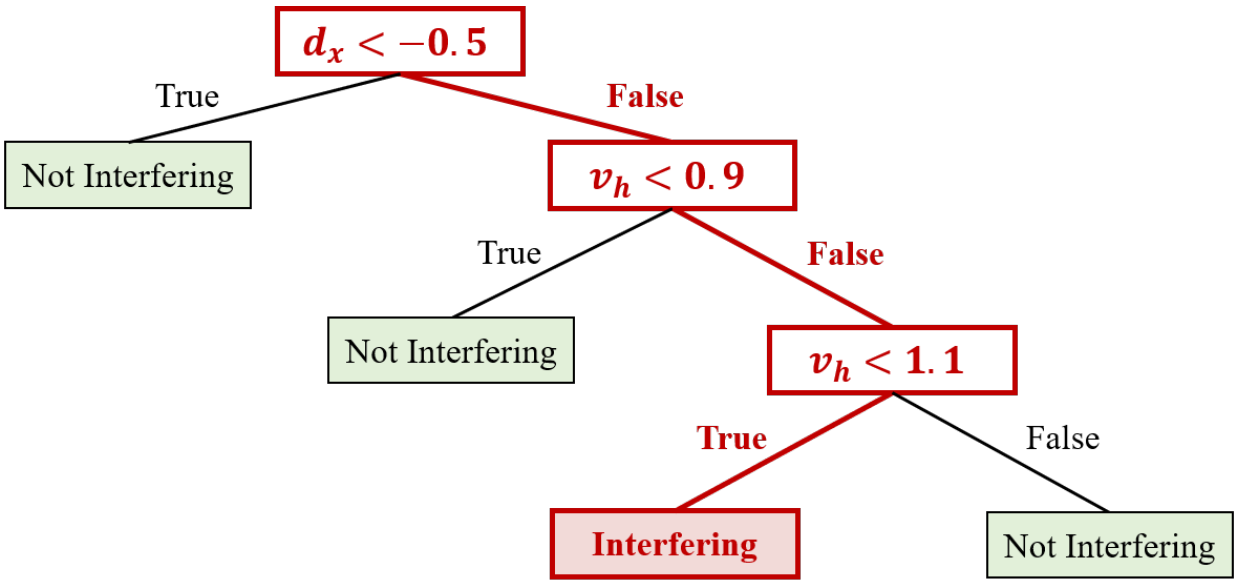


Figure 4.5: Prediction and explanation decision tree $\mathcal{T}_p^h$.

The attributes in this particular example are the following:

$$\boldsymbol{\alpha}_h(t) = \begin{bmatrix} d_x & d_y & \theta & v_h & v_r \\ 0 & -3 & \pi/4 & 1.0 & 0.8 \end{bmatrix}$$

The prediction is $\mathcal{P}_h(t) = \lambda$ (interfering), indicated by the red path in the figure. Through (4.15), the following explanation is obtained:

$\mathcal{E}_h(t) =$ ***Interfering*** because: $\{d_x > -0.5 \wedge 0.9 < v_h < 1.1\}$

Through (4.16), the following counterfactuals are then obtained:

$\boldsymbol{\mathcal{C}}_h(t) =$ ***Not Interfering*** when:

$\{d_x < -0.5\} \vee$

$\{d_x > -0.5 \wedge v_h < 0.9\} \vee$

$\{d_x > -0.5 \wedge v_h > 1.1\}$

4.4.4 Counterfactual Analysis and Priority-based Correction

When the VP planner is predicted to interfere with an actor, the counterfactuals of the DTs tell which attributes, when changed, can revert the prediction. The robot, however, cannot change

attributes that are related to actor motion, and can only control its maximum target speed, v_r, which is used to obtain $\boldsymbol{u}_{att}$ (4.1). To find corrective target speeds for the robot, we build a set of secondary trees, $\mathcal{T}_c^h$ and $\mathcal{T}_r^h$, in which we fix all attributes except v_r and find similar states from $\mathcal{S}$ as done in (4.14), creating a new set $\boldsymbol{s}_c \subset \mathcal{S}$ (note that $\boldsymbol{s}_h \subset \boldsymbol{s}_c$). In this way, new DTs are still relevant to other local attributes but include varied robot speed, enabling DTs to find from within the finite state space $\mathcal{S}$ via (4.16) a set of non-interfering target speeds, $\boldsymbol{v}_{corr}^h$.

In Fig. 4.6, we show the correction tree associated with the example introduced in Section 4.4.3.

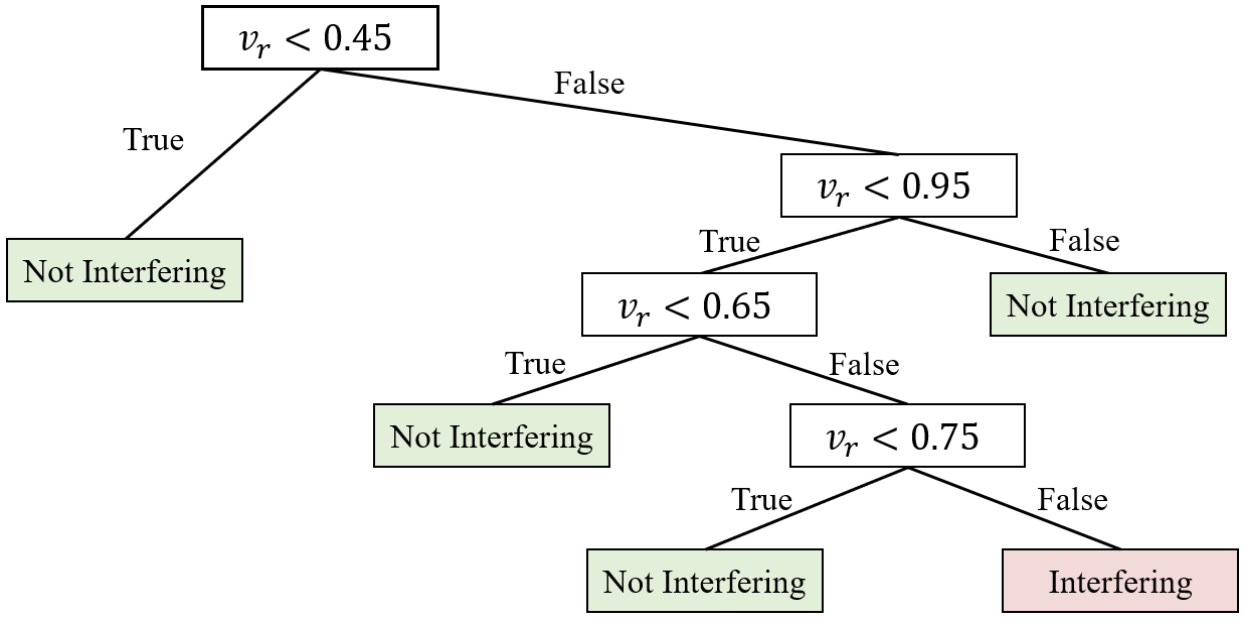


Figure 4.6: Correction decision tree $\mathcal{T}_c^h$.

The robot was moving at a desired speed of $v_r = 0.8$. Through (4.16), we obtain the following set of counterfactuals:

$\boldsymbol{\mathcal{C}}_h(t) = \boldsymbol{Not\ Interfering}$ when:

$\{v_r < 0.45\} \vee$

$\{0.45 \leq v_r < 0.65\} \vee$

$\{0.65 \leq v_r < 0.75\} \vee$

$\{v_r > 0.95\}$

In this example, the set of non-interfering speeds from our state space $\mathcal{S}$ is $\boldsymbol{v}_{corr}^h = [0\ 0.1\ 0.2\ 0.3\ 0.4\ 0.5\ 0.6\ 0.7\ 1.0]$

While each target speed within $\boldsymbol{v}_{corr}^{h}$ may satisfy the conditions for a non-interfering case, some may result in better or worse performance, depending on the priority behavior of the robot (Problem 2).

The optimal target speed for given priority parameters is computed by comparing the output of non-interfering speeds, $v \in \boldsymbol{v}_{corr}^{h}$, in the reward tree of actor h, as follows:

$$v^* = \arg\max_{v} \mathcal{T}_r^h(\boldsymbol{v}_{corr}) \tag{4.17}$$

This is then set as the maximum target speed for the attractive input, discussed in Sec. 4.2.

Shown in Table 4.1 is the output of the reward tree with the following parameters $\boldsymbol{\beta} =$ [0.25 0.25 0.25 0.25], taken from the running example in the previous sections (Figs. 4.5, 4.6). The non-interfering speeds are compared using the predicted reward, and through (4.17), we obtain that $v^* = 0.4$m/s.

Table 4.1: Rewards from $\mathcal{T}_r^h$

$v_i \in \boldsymbol{v}_{corr}$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0
Reward	0.601	0.690	0.744	0.747	0.801	0.771	0.732	0.554	0.619

4.4.5 Extension to Multiple Actors

The set of non-interfering speeds for multiple surrounding actors is the intersection between individual sets. Let v_{corr}^{h} represent a set of non-interfering speeds for a single actor h within the sensing range of the robot. Then the intersection of sets of all actors is represented as follows:

$$\hat{\boldsymbol{v}}_{corr} = \bigcap_{h} \boldsymbol{v}_{corr}^{h} \tag{4.18}$$

The set of combined non-interfering speeds, $\hat{\boldsymbol{v}}_{corr}$ includes speeds for which $\mathcal{P}_h = \neg\lambda \; \forall h = 1, \ldots, N_h$, where N_h is the number of actors within the sensing range of the robot. Then the optimal target speed for all actors is found similarly as in (4.17), but now includes summing the rewards:

$$v^* = \arg\max_{v} \sum_{h=1}^{N_h} \mathcal{T}_r^h(\hat{\boldsymbol{v}}_{corr}) \tag{4.19}$$

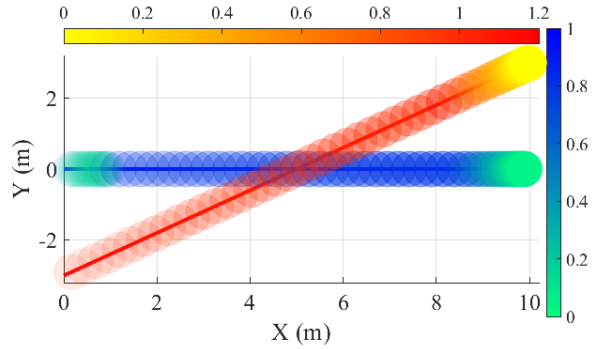
There may, however, exist cases in which no global solution can be found, that is $\hat{\boldsymbol{v}}_{corr} = \varnothing$, due to particularly dense conditions and contradicting corrections. In these cases, the system reverts to a fail-safe mode, in which the robot moves at a very low target speed with our VP planner. We treat this as fail-safe because, in addition to our assumption that actors are non-hostile, assigning a low speed has been shown to effectively communicate to actors that the robot is accommodating [57] giving the actors more time to react and perform avoiding maneuvers. However, when actors take upon the burden of avoidance, interfering conditions are often violated. These cases, as with all observations, are recorded and probabilities are updated through (4.8) and (4.9) at runtime, refining predictions so that the robot can learn to avoid getting into a similar interfering state in the future.

4.5 Simulations

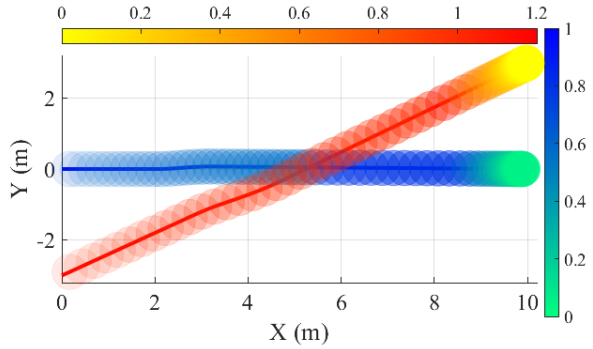
In this section we provide several MATLAB simulations to evaluate and compare our approach with other methods. Simulation training included a robot equipped with the previously described VP planner with target speeds in the interval $v_r = [0.0, 1.0]$m/s, discretized at 0.1m/s increments. The desired speed was set to $v_{des} = 0.8$m/s. In the training, the robot performed its trajectory in the presence of one actor also running the aforementioned VP planner, which had different initial and final positions in each trajectory with a nominal speed of $v_h = 1$m/s.

In the first simulation, we show the outcome of the running example presented throughout this chapter so far (Figs. 4.5 and 4.6), in which a robot with a 5m sensing radius is navigating in the presence of one actor and has a goal at $(10, 0)$m. The distance and deviation thresholds are $\delta_s, \sigma_h, \sigma_r = 1$m. Fig. 4.7 compares our approach under two different priority behaviors, and contrasts results with the standard VP planner.

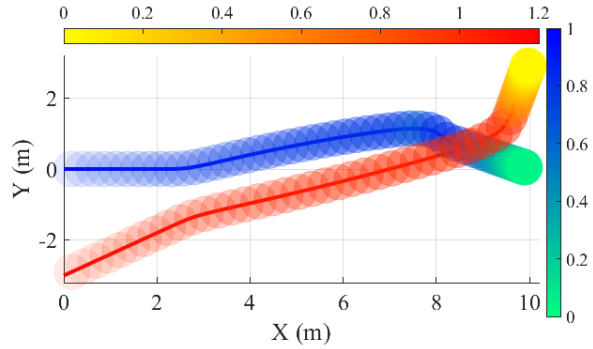
In Fig. 4.7(a) the robot prioritizes minimizing actor deviation by using priority reward parameters $\boldsymbol{\beta} = [0.2\ 0.1\ 0.6\ 0.1]$. The resulting trajectory consists of a very low target speed of 0.2m/s, and no deviations. In Fig. 4.7(b) we repeat the same case study, but the robot is now tasked to prioritize maintaining its desired speed ($\boldsymbol{\beta} = [0.2\ 0.1\ 0.1\ 0.6]$). As can be noted in Fig. 4.7(d) the robot maintains a speed of 0.6m/s, closer to the desired 0.8m/s, the maximum actor deviation (Fig. 4.7(e)) is 0.17m, and the robot's is 0.05m due to the underlying VP planners but interference conditions (4.4) are not violated. Fig. 4.7(c) shows the VP planner without our approach. The robot maintains its



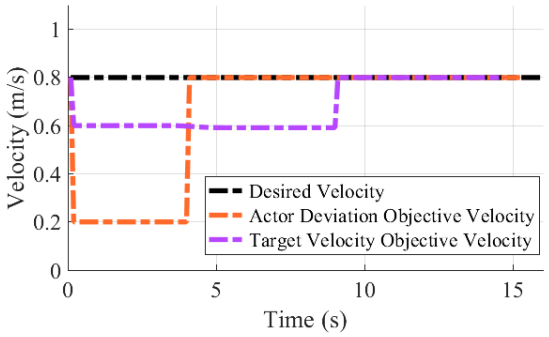
(a) Trajectories minimizing actor deviation.



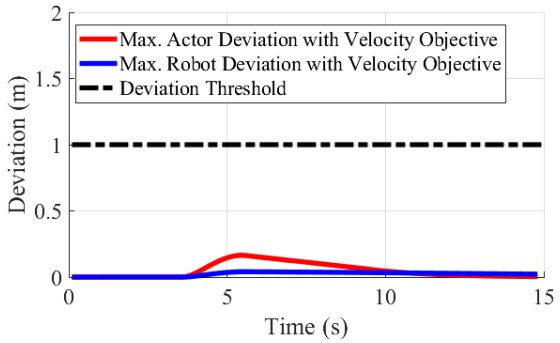
(b) Trajectories minimizing deviation from robot desired velocity.



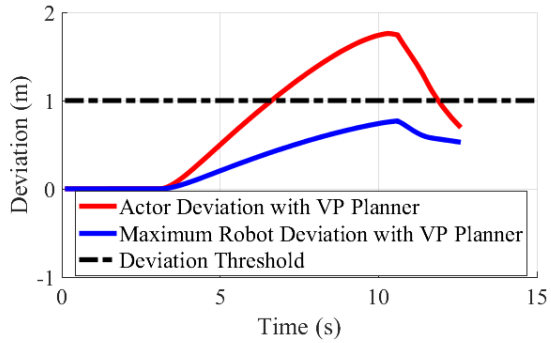
(c) Trajectories with a standard VP planner



(d) Speed comparison results between objectives.



(e) Path deviations with the desired speed objective



(f) Path deviations with the VP planner

Figure 4.7: Trajectories and results of robot (cool colors) and actor (warm colors) comparing the presented approach with different objectives and the standard VP planner without our approach. Velocities are indicated by the color-bars within the trajectories.

desired speed of 0.8m/s, and the maximum actor deviation is 1.76m (Fig. 4.7(f)), which violates the interference threshold, showing that our approach was effective in correcting the robot's interfering behaviors. The presence of such interference with only the VP planner indicates that our approach is effective in predicting and finding the appropriate target velocities that proactively eliminate an interference that would have been caused in the future.

We also extensively tested our approach in dense multi-actor environments, where the robot traverses 12m, sharing an environment with 10 simulated actors running a standard VP planner to avoid the robot. We compare our approach under different priorities with two widely-used planners: ORCA [100] and DWA [36], which is the standard planner on ROS-enabled systems. The results over 10 trials, in which the robot navigates through a total of 100 actors, are shown in Table 4.2.

Table 4.2: Results from comparative simulations.

	Success Rate (%)	Average Actor Deviation (m)	Average Max Robot Deviation (m)	Target Speed Deviation (m/s)	Average Added Time (%)
Standard VP Planner	62%	0.792	1.311	0	*21%*
Neutral Priorities	87%	0.351	0.407	0.598	45%
Maintaining Desired Speed	86%	0.356	0.463	**0.507**	**41%**
Minimizing Actor Deviation	**89%**	**0.298**	**0.283**	0.702	49%
ORCA	75%	0.433	0.981	0.544	41%
DWA	44%	0.361	1.511	0.821	64%

The success rate is defined as the proportion of total actors that were interfered. When prioritizing minimal actor deviation, our approach outperforms others with an 89% success rate. However, we find that results are similar under different priorities, since the solution set, $\hat{\boldsymbol{v}}_{corr}$, is often limited in dense environments. The actor deviation objective is the most conservative, resulting in lower speeds and higher added time of 49%, when compared to the time an unobstructed path to the goal would have taken. The standard VP planner has the lowest added time, but has a success rate of 64%. In the cases in which our approach fails, we found that the robot was negotiating with dense crowds ($N_h > 6$) within its sensing range and no solutions were available.

The presented approach outperforms ORCA in all metrics, but we do note that ORCA outperforms the standard VP planner, because ORCA expects surrounding actors to follow the same algorithm and perform reciprocal avoiding behaviors. While actors using VP planners are not exactly finding reciprocal velocities, they are partially satisfying ORCA expectations by avoiding the robot. DWA, on the other hand, is searching for a set of admissible velocities given the occupancy

in the robot's sensing range. Since this approach is not considering the future motion of dynamic actors, we observe a larger robot deviation. This, in turn, increases the added time to the robot trajectories, and results in a poorer success rate, when compared to other approaches.

4.6 Experiments

The proposed approach was also validated experimentally using unmanned ground (UGV) and aerial (UAV) vehicles in indoor environments. HMM and DT operations were implemented in MATLAB and interfaced with ROS through the ROS Toolbox, and executed at 10Hz. For all experiments, the distance threshold $\delta_s = 1$m and the deviation thresholds $\sigma_r, \sigma_h = 0.5m$. The training speeds were set to 0.2, 0.4, and 0.6m/s while $v_{des} = 0.6$m. The robot starts at (-2.5,0)m and is tasked to reach a goal at (2.5,0)m. We depict the trajectories from experiments training in Fig. 4.8, where robot trajectories are in blue and actor trajectories are in red. This training dataset is included as "dtreevp_dataset.7z" in the GitHub repository in [77].

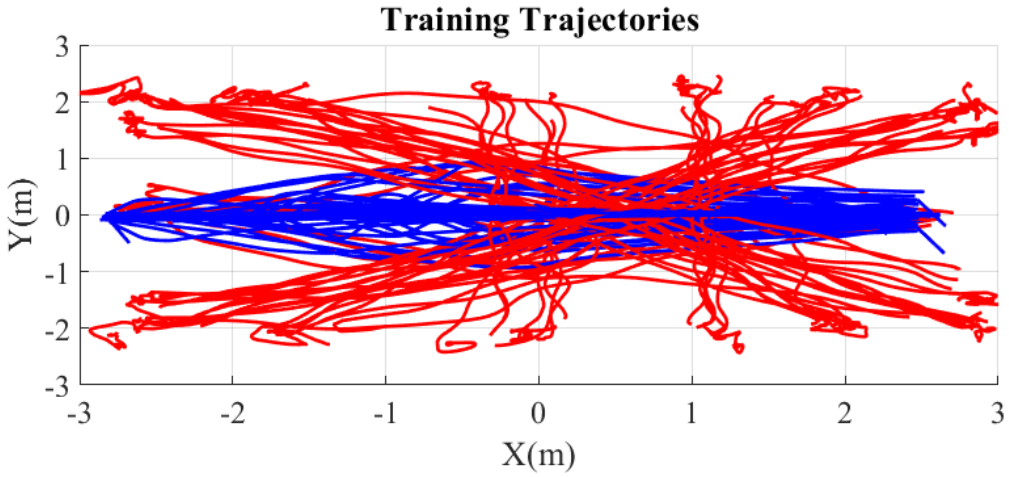
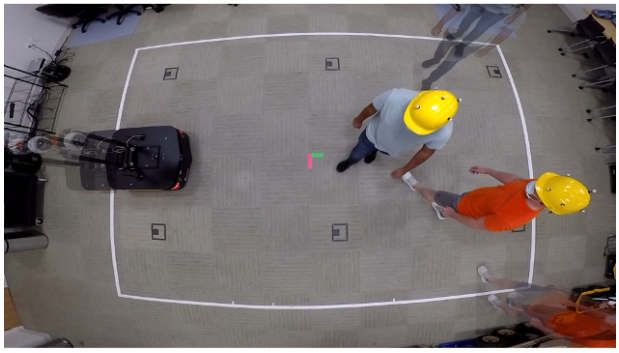
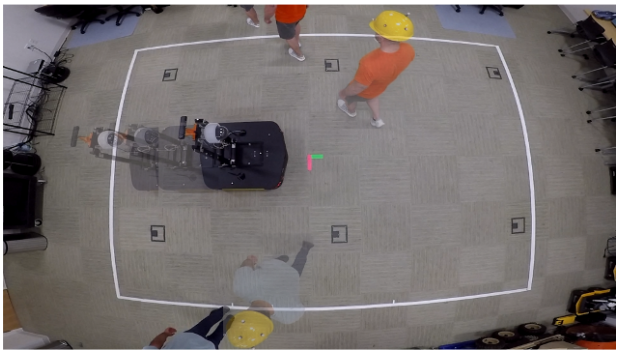


Figure 4.8: Experiment Training Trajectories.

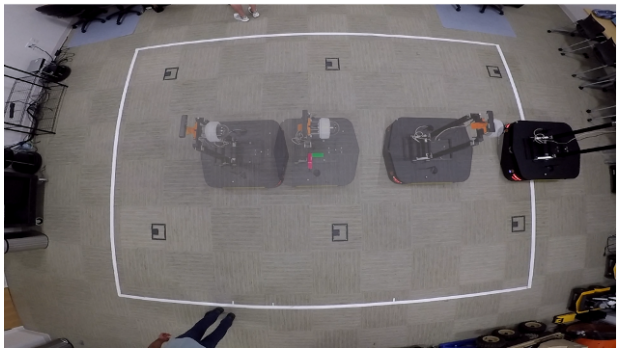
Fig. 4.9 shows results from ground vehicle experiments on a Clearpath Robotics Ridgeback Omnidirectional Industrial Robot inside our lab tracked by a VICON motion capture system (MOCAP). With the more conservative priorities (Figs 4.9(a-d)), the robot reduces its target speed to $v_r = 0.2$m/s and no deviation on either the robot or the actors paths is detected. When the robot prioritizes maintaining desired speed, our approach only reduces to $v_r = 0.4$m/s with a minor



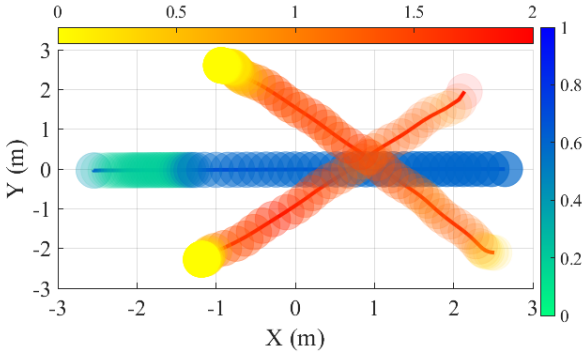
(a) Robot slows down after predicting interference



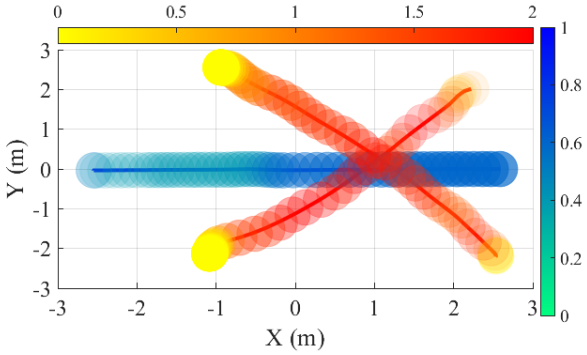
(b) Robot no longer predicts interference with actors



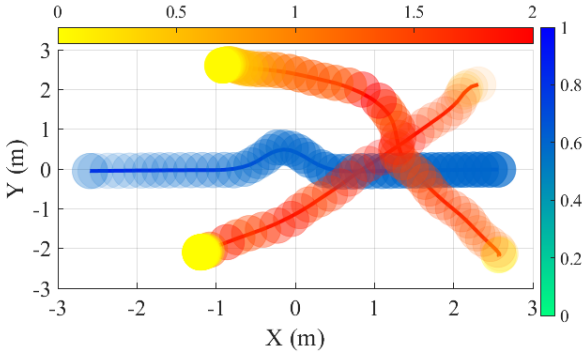
(c) Robot returns to desired velocity to reach its goal



(d) Trajectories minimizing actor deviation.



(e) Trajectories minimizing velocity deviation.



(f) Trajectories with a standard VP planner.

Figure 4.9: Ground vehicle experiment trajectories and snapshots of robot (cool colors) and actor (warm colors) comparing the presented approach with different objectives and the standard VP planner without our approach.

deviation from its path of 0.04m. Under the standard VP planner, the robot deviates 0.53m, and the actor deviates ∼1m from the desired path, violating the 0.5m threshold.

We also successfully tested the applicability of our approach outside MOCAP settings. We deployed our technique on the same UGV using only the on-board RGB-D camera for person detection and Lidar sensors for localization in the presence of 3 actors.

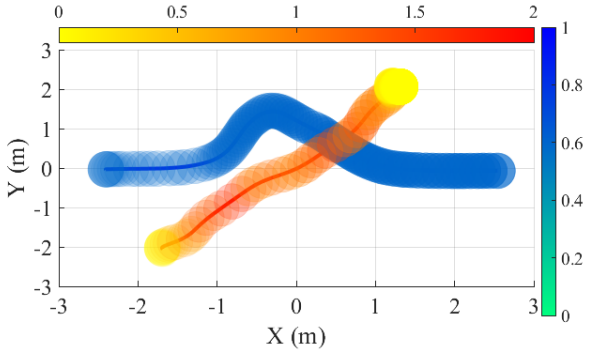
In the UAV experiment, we replicated a failure case shown in simulation (Fig. 4.7(c)) with an aerial vehicle. We used the DJI Tello mini drone in the same lab environment, with the same goal and parameters. Fig. 4.10 contains trajectories and snapshots from these experiments. The UAV with the standard VP planner deviates 1.45m, taking a longer path to reach the goal. With our approach, the robot adapts its speed and deviates only 0.11m. Notably, the UAV reaches the goal 2s *faster*, despite slowing down to avoid interference.

4.7 Discussion

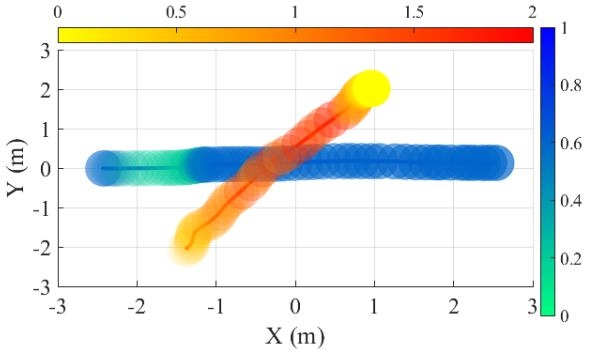
In this chapter, we have presented a novel approach that leverages virtual physics planning by compensating for local minima failures that cause interference with dynamic actors. We generate proactive non-interfering behaviors between a robot and nearby dynamic actors through a decision tree-based explainable monitor. A HMM-based probability model was also introduced to enable DTs to update and improve predictions and planning quickly at runtime. In particular, this update is more efficient when compared to the data-driven updating approaches presented in the previous chapters. Finally, we presented a reward-based method to trigger different non-interfering behaviors given the priority objectives of the robot. The main benefits of our approach are that it leverages the efficiency and low overhead of VP planning methods and provides human interpretable explanations when interference is detected. This framework scales well to real-world settings as demonstrated by the different experiments conducted with different robots with different sensing capabilities.

4.7.1 Limitations and Future Directions

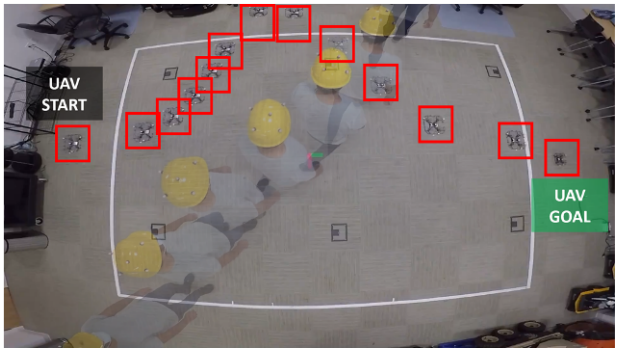
Dealing with multiple actors still remains a challenge in dense situations where a global solution doesn't exist. While in this work, the robot switches into a fail-safe mode, we note that there can be situations in which a sequence of actions may be able to create non-interfering behavior while



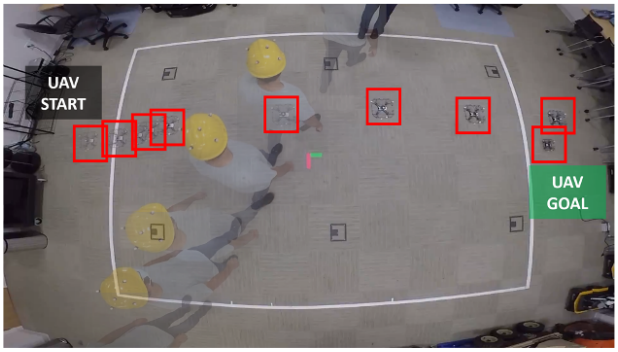
(a) VP planner trajectories.



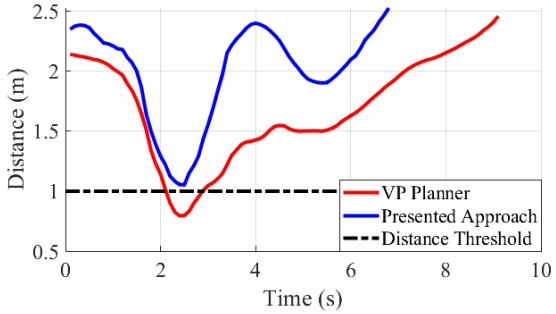
(b) Trajectories with our approach.



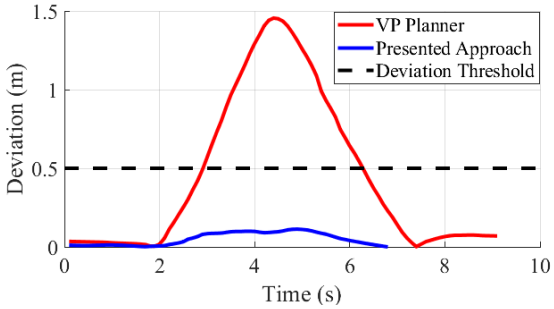
(c) Snapshots for the experiment with just a VP planner.



(d) Snapshots for the experiment with our approach.



(e) Distance between agents.



(f) Deviation of robot path.

Figure 4.10: UAV experiment trajectories, snapshots, and results.

maintaining a certain level of performance of the robot. Another consideration is that this approach considers human behavior with only data-driven training examples. Pairing a deterministic dynamic model with data-driven predictions, on the other hand, could be more powerful for estimating the future behaviors of actors and the robot. Furthermore, once actor dynamics are considered, and interactions between actors are modeled, it may be the case that considering all actors equally may not be prudent for finding an appropriate sequence of actions to correct robot behaviors. In future work, we plan to expand on these thoughts and further investigate how we can consider crowd interactions and prioritize certain actors in dense environments. We investigate these ideas in the next chapter of this dissertation: we consider dynamic models of the robot and humans within a model predictive control (MPC) framework and model interactions between humans to find out how we can mitigate some of the aforementioned issues.

Chapter 5

Attention-aware Robot Social Planning

In this chapter, we expand on our previous work in the social navigation case study to improve behaviors as the number of actors in the robot's sensing range increases. Our previous approaches have treated each individual separately, and composed together the individual corrective behaviors to plan robot motion, but we note that considering the interactions between individual actors in a crowd setting can help determine whether the robot will violate the previously considered interference conditions. In this way, the problem we address in this chapter is the same: the robot should avoid causing a burden on humans in the environment, while taking into account human-human interactions within the crowd. To this end, we leverage the concept of attention in crowd navigation, in which the robot predicts which actors it most needs to pay attention to in order to move in a socially acceptable way. We formulate a model predictive control (MPC) framework for robot motion that includes the dynamic model of the actors in the environment, and we outline the computational and feasibility benefits of limiting the actors the robot pays attention to. To achieve this, we design a deep neural network (DNN) to predict attention for pairs of humans at runtime. Notably, we provide a composition method, through which we are able to extend the predictions to scenarios with multiple actors while limiting training and DNN predictions to pairs of humans. We show through extensive simulations that our approach is able to leverage pairwise interactions to minimize the actors the robot considers while also maintaining a low level of overall interference with all the actors. This work is in the final stages of preparation to be submitted to the Robotics and Automation Letters.

5.1 Introduction

When mobile robots operate in real-world scenarios they may have to deal with several actors at a time (e.g., in urban scenarios, airports, workspaces, etc.). However, including every actor as a constraint to an optimal planning controller or training with any number of actors may be computationally expensive, making navigating through these environments challenging for mobile robots.

As humans, however, we find that we can often navigate such environments in a non-intrusive way with relative ease. But how are we able to succeed in large crowds? Researchers in psychology and cognitive science [60] have shown that we learn, through our experiences, to focus our attention on the most important things in a given scenario, since we cannot pay attention to too many different things at a time. In the case of crowd navigation, we are able to identify the people within the crowd that we believe are most likely to be involved in some future interaction, and we primarily devote our attention to the behaviors of these actors when we plan our motion. At the same time, when deciding which actors to pay attention to, we implicitly encode the collective impact of all other neighboring actors, resulting in behaviors that perhaps pay attention to only a few actors, but seamlessly navigate through all actors. Importantly, we make these decisions without necessarily making sophisticated predictions of the paths of the actors – we tend to make simplifying assumptions about what people will do [13] – and we can provide physical context and reasoning about how we use attention to plan our motion. For example, we may not necessarily pay attention to the closest people to us in a crowd because we may interact first with a person who is further away, but advancing towards us very quickly. Bringing the context back to robot navigation, it is further beneficial to identify important actors, as considering all visible actors in a crowd equally can be too computationally expensive, or may make it very difficult to find a solution that satisfies conditions for all actors, as we observed in previous chapters of this dissertation. On the other hand, if a robot could identify important actors and plan motion in a similar way as humans do, it could generate similarly natural and seamless motion in dense crowds without causing computational issues. To summarize, the main challenge we address in this chapter is:

- How to identify and only consider the most important actors while still capturing the impact

of less important actors in order to avoid interfering with all actors in the robot's sensing range.

Figure 5.1: The desired effect of our attentive social planning approach. Only the necessary actors (colored in purple) are modeled to ensure the robot can find a socially acceptable path through the environment without interfering with the behaviors of other actors (in red).

To this end, we propose an approach that leverages a deep neural network (DNN) along with a novel composition module that takes in observable states of nearby actors (relative positions, velocities), and determines which actors in the scene are most relevant to the robot's planning. Notably, our composition module enables us to train with pairs of actors, and at runtime, combine pairwise predictions in crowds with multiple (i.e., more than a pair) actors. In this work, pairwise interactions are leveraged since it has been shown in statistical and biological physics-based modeling [22, 66] that pairwise interactions can be used to effectively approximate crowd behaviors. Furthermore recent work in robot collision avoidance [19] has leveraged pairwise interactions within a reinforcement learning framework to effectively avoid colliding with interacting agents. For robot planning in our work, we leverage model predictive control (MPC) that takes into account the dynamic model of the robot and actors to cast forward future predictions and avoid actors safely and proactively while enabling the robot to reach its goal. Importantly, MPC works by solving an

optimal control problem, which is more precise than end-to-end learning-based planners, and thus we do not need to train how to obtain controls for the robot. However, it should be noted that our approach can work for any other controller that can plan motion in the presence of multiple dynamic obstacles, since the main contribution of our approach (prediction and composition) is to provide context for these controllers about which actors to avoid. Fig. 5.1 demonstrates the desired effect of our approach, in which the robot is able to quickly find a path through a busy, dynamic environment by limiting actors to whom it pays attention.

5.2 Preliminaries

In this section, we describe the premises of our robot crowd navigation case study. Let us consider that a mobile robot navigating through a crowd is equipped with a Model Predictive Controller (MPC). To plan robot behaviors, the MPC makes predictions about the future positions of both the robot and the actors to find control inputs that minimize a certain cost function. Thus, the MPC must contain information about the dynamics of the robot and actors, as well as constraints so the robot plans a safe path to the goal.

5.2.1 Robot and Human Dynamic Models

First, we address the modeling aspect required to use an MPC. The robot follows a standard non-holonomic unicycle model [29], where the state of the robot $\boldsymbol{x}_r = [x_r, y_r, \theta_r]^\top$ consists of x and y positions and orientation θ. The input, $\boldsymbol{u}_r = [v_r, \omega_r]^\top$, to the robot contains linear and angular velocities. The equations of motion for the robot are as follows:

$$\begin{aligned} \boldsymbol{u}_r &= [v_r, \omega_r]^\top \\ \dot{x} &= v_r \cos\theta \\ \dot{y} &= v_r \sin\theta \\ \dot{\theta} &= \omega_r \end{aligned} \tag{5.1}$$

The humans, in this work, are assumed to use the well-known and widely-used social force model [38], which has shown to effectively approximate the behaviors of humans in crowds [33].

The social force model is based on psychological forces that enable pedestrians to move to their desired goals, while keeping a comfortable distance from others that share the environment. In this work, we model the state of human i as $\boldsymbol{x}_i = [x_i, y_i]$ and the social forces are modeled as velocities $\boldsymbol{u}_i = [v_{i_x}, v_{i_y}]$ that move the humans to their goals, which are unknown to the robot. Then the equations of motion for human i are defined as follows:

$$\begin{aligned} \boldsymbol{u}_i &= [v_{i_x}, v_{i_y}] \\ \dot{x_i} &= v_{i_x} \\ \dot{y_i} &= v_{i_y} \end{aligned} \tag{5.2}$$

The specific components of the input $\boldsymbol{u}_i$ include the goal force, actor avoidance force, and obstacle avoidance force. In the following formulations, time t is omitted for ease, but each of these forces evolves with time with the changing states of the human i and the surrounding actors including the robot j. The goal force for a human i is computed as follows:

$$\boldsymbol{u}_{\boldsymbol{g}_i} = k_g(\boldsymbol{x}_{\boldsymbol{g}_i} - \boldsymbol{x}_i) \tag{5.3}$$

where k_g is a positive constant that affects the strength of the force and $\boldsymbol{x}_{\boldsymbol{g}_i}$ is the goal for human i. The actor avoidance force is a repulsive force that drives a human away from other moving actors and is computed as:

$$\boldsymbol{u}_{ij} = a_i \exp(\frac{\delta_h - d_{ij}}{b_i}) \vec{\boldsymbol{d}}_{ij} \tag{5.4}$$

where a_i and b_i are positive constants that affect the strength and effective range of repulsion from each of the other actors. The distance between actors i and j is denoted by d_{ij} the desired distance between humans, is given by δ_h, and $\vec{\boldsymbol{d}}_{ij}$ is a unit vector which directs human i away from actor j. For ease, we consider the desired distance between all pairs of actors to be equal, but this is not a necessary condition for the approach presented in this work. Finally, the obstacle repulsion force is computed similarly to (5.4):

$$\boldsymbol{u}_{io} = a_o \exp(\frac{r_o - d_{io}}{b_o}) \vec{\boldsymbol{d}}_{io} \tag{5.5}$$

where a_o and b_o are the positive constants that now represent obstacle avoidance force, d_{io} is the

distance between human i and each obstacle o in the environment, r_o represents a minimum distance to maintain from obstacles, and the unit vector in this expression moves humans away from static obstacles.

Then the total force applied to each human i at any time can be given by the sum of the aforementioned forces:

$$\boldsymbol{u}_i = \boldsymbol{u}_{\boldsymbol{g}_i} + \boldsymbol{u}_{ij} + \boldsymbol{u}_{io} \tag{5.6}$$

This model is utilized to make predictions about future states of dynamic actors in the MPC framework, which is described in the next section.

5.2.2 Model Predictive Controller Formulation

MPC is utilized in this work because it is a deterministic baseline controller that can predict future values of some optimal control problem (OCP), and find inputs that minimize these future values over a future horizon N. The MPC formulation consists of a cost function J to minimize at each time step, and constraints that be must respected when minimizing this cost. We use the standard MPC formulation with the following cost function:

$$\begin{aligned} J =& (\boldsymbol{x}_r(t+N) - \boldsymbol{x}_{\boldsymbol{g}_r})^\intercal \boldsymbol{Q} (\boldsymbol{x}(t+N) - \boldsymbol{x}_{\boldsymbol{g}_r}) + \\ & \sum_{k=1}^{N-1} (\boldsymbol{x}_r(t+k) - \boldsymbol{x}_{\boldsymbol{g}_r})^\intercal \boldsymbol{Q} (\boldsymbol{x}_r(t+k) - \boldsymbol{x}_{\boldsymbol{g}_r}) + \\ & \boldsymbol{u}_r(t+k-1)^\intercal \boldsymbol{R} \boldsymbol{u}_r(t+k-1) \end{aligned} \tag{5.7}$$

where N is the prediction horizon, $\boldsymbol{x}_r(k)$ is the robot state, which consists of position and orientation, $[x\ y\ \theta]^\top$, for the kth prediction step, $\boldsymbol{x}_{\boldsymbol{g}_r}$ is the goal state, $\boldsymbol{u}_r(k)$ is the kth control input computed by the MPC, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the cost weighting matrices for state and control input reference

tracking, respectively. Then, the optimal control problem can be formulated as:

$$\underset{\boldsymbol{u}_r(0),\ldots,\boldsymbol{u}_r(N-1)}{\arg\min} \quad J(\boldsymbol{x}_r(0), \boldsymbol{u}_r(0), \ldots, \boldsymbol{u}_r(N-1)) \tag{5.8}$$

subj. to:

$$\boldsymbol{x}_r(t+k+1) = \boldsymbol{f}(\boldsymbol{x}_r(t+k), \boldsymbol{u}_r(t+k)),\ \forall k = [0, N] \tag{5.9}$$

$$\boldsymbol{x}_i(t+k+1) = \boldsymbol{h}(\boldsymbol{x}_i(t+k), \boldsymbol{u}_i(t+k)),\ \forall k = [0, N],\ \forall i \in H(t) \tag{5.10}$$

$$||\boldsymbol{x}_r(t+k) - \boldsymbol{x}_i(t+k)|| > \delta_s,\ \forall k = [0, N],\ \forall i \in H(t) \tag{5.11}$$

$$\boldsymbol{u}_r(t+k) \in \mathcal{U}(t+k),\ \forall k = [0, N-1] \tag{5.12}$$

$$\boldsymbol{x}_r(t+k) \in \mathcal{X}(t+k),\ \forall k = [0, N] \tag{5.13}$$

where (5.9) is a constraint on robot dynamics, (5.10) is a constraint on human dynamics, and (5.11) is a constraint on the distance between the robot and humans. In these constraints, $\boldsymbol{f}(\boldsymbol{x}_r, \boldsymbol{u}_r)$ represents the robot dynamic model (5.1), $\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i)$ represents the human dynamic model (5.2), $H(t)$ is the set of humans within the robot's sensing range, and δ_s is a minimum desired distance between robot position $[x_r,\ y_r]$ and human state $\boldsymbol{x}_i$, which can be drawn from the Proxemics model for pedestrian personal space [32]. The Proxemics model is a psychologically motivated model that suggests comfortable distance between humans is different than that between a human and a robot; typically a robot should stay further from humans than other humans, and we account for this in our constraint by setting $\delta_s > d_{ij}$.

The feasible region for robot inputs (5.12) is defined by $\mathcal{U}$, and $\mathcal{X}$ is the feasible region for the robot state (5.13), given environmental constraints such as the position of walls and static obstacles. Note that the human positions are not explicitly modeled into the feasible region $\mathcal{F}$ but (5.11) instead restricts the feasible region for the robot indirectly through the desired distance requirement.

In the context of this work, we are most concerned with (5.11). While an MPC provides a deterministic and safe baseline planner, this type of constraint can cause two main issues for robot motion planning:

- Including all humans in the environment as individual constraints that must be respected increases the computation time of the MPC, making it very slow to plan robot behaviors.

- As the number of humans in H grows, assuming all are formulated as individual constraints, the robot's feasible region $\mathcal{X}$ becomes more restricted, making it difficult for the MPC to find a feasible solution.

Both of these issues are connected to modeling and trying to avoid all humans in H throughout each step in the prediction horizon N, and these types of issues could cause the robot to get trapped or "frozen" [97] in the environment, and place the burden on surrounding humans to deviate in order to accommodate the robot. Shown in Fig. 5.2 is the effect of adding more constraints on MPC computation time, where it is clear that adding constraints increases the computational cost of the MPC. The challenge then becomes limiting negative effects of (5.11) in a way that the MPC is able to quickly find a solution that still minimizes causing burden on the nearby humans.

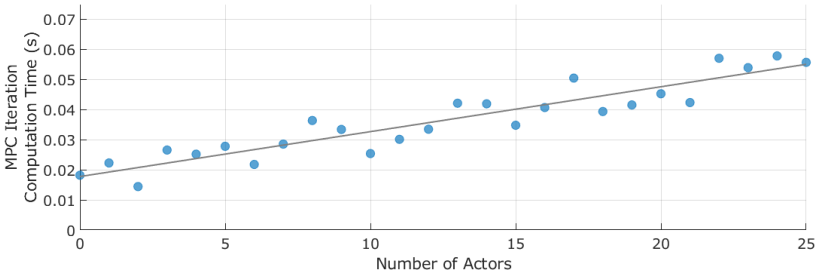


Figure 5.2: MPC computation times with different numbers of actors included as constraints.

5.3 Problem Formulation

Consider a mobile robot equipped with the aforementioned MPC that is tasked to navigate to a goal in a shared environment with other dynamic actors. With the premises presented in Sec. 5.2, we would like to directly predict the smallest set of actors $H_a \subseteq H$ the MPC needs to consider to plan robot motion that minimizes interfering with the motion of all humans H in the environment. Formally, the problem is:

Problem 5.1 (Attention-aware Social Planning) *Given the control policy given in* (5.8)*, derive a strategy to find the minimum number of actors* $H_a \subset H$ *to include in constraints* (5.10) *and*

(5.11) *such that the overall actor deviation from their desired path is minimized at all times:*

$$\begin{aligned} &[\min J \quad subj.\ to:\ H_a] \rightarrow d^* \\ &[\min J \quad subj.\ to:\ H] \rightarrow d \\ &\quad and \quad d^* \leq d \pm \epsilon \end{aligned} \tag{5.14}$$

where d^* is the overall deviation if the MPC optimization considers only H_a, and d is the deviation when considering the entire set H, and ϵ is a small error term.

5.4 Approach

In this section, we describe our framework for attentive MPC-based social planning. Our framework consists of an offline training phase, through which a prediction model is built to be leveraged in an online deployment phase as depicted by the architecture in Fig. 5.3.

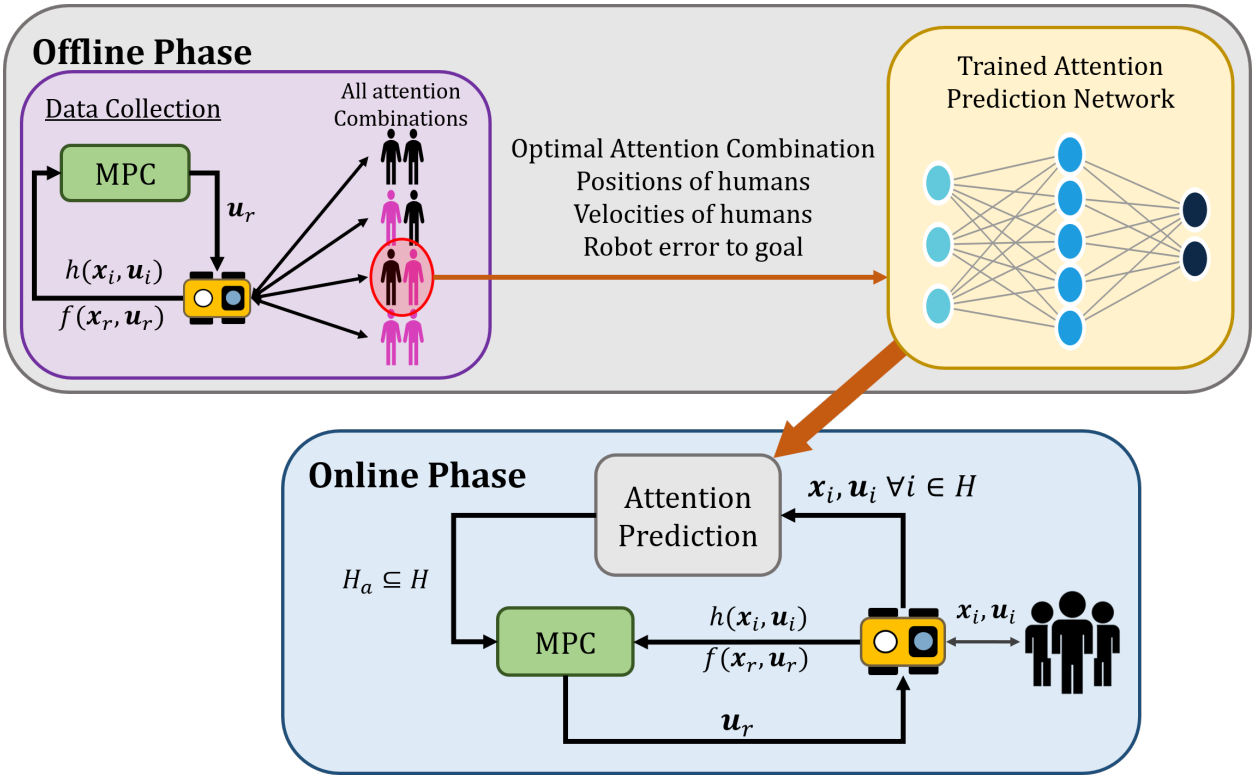


Figure 5.3: Block diagram of our attention-aware crowd navigation approach.

In the offline stage, a robot in the presence of humans performs go-to-goal navigation tasks using an MPC, which takes into account both robot dynamics $f(\boldsymbol{x}_r, \boldsymbol{u}_r)$ and human dynamics $h(\boldsymbol{x}_i, \boldsymbol{u}_i)$. Different attention combinations, indicated by the magenta and black human icons are tested. The optimal combination H_a is the label, and the associated predictors include the positions and velocities of the humans, $\boldsymbol{x}_i, \boldsymbol{u}_i,\ \forall i \in H$, and the robot's error to goal, which captures, at high level, the desired behaviors for the robot to reach its goal. The training is repeated with different configurations of humans and robots. Then, the predictors and labels are used to train a prediction model that will be used online to evaluate which actors the robot should pay attention to in order to avoid interfering with all actors in the shared environment. At runtime, the robot observes the positions and velocities of nearby actors and utilizes the trained attention prediction network to predict which actors $H_a \subseteq H$ to include as constraints for the MPC, which then computes and sends inputs to the robot. This procedure is repeated at runtime since the environment is highly dynamic and new actors can appear and leave the robot's sensing range. In the following sections, we describe in detail each part of the proposed approach.

5.4.1 Training Details

The training dataset used to build the attention prediction network consists of a robot using the MPC presented in Sec. 5.2 to perform go-to-goal operations in the presence of multiple moving actors with different attention combinations. In our previous work [78], we have modeled human-robot interactions individually and composed them together to make predictions for multiple actors at runtime. While this type of composition can show good results [79], it neglects the fact that humans in the environment are also interacting with each other, and when humans alter each other's behaviors, it affects how the robot should plan its own behaviors. Thus, we include multiple moving humans in the training samples, and model human-human interactions in crowds using the social force model presented in Sec. 5.2.

Specifically, in the proposed approach, we leverage training data consisting of the robot and a pair of humans (i.e., a set of two different humans with different initial positions and velocities). We leverage pairwise interactions here since it is a minimal representation through which human-human interactions can be captured and since these pairwise interactions have been shown to accurately represent subsets of interactions between multiple actors [22, 66, 19]. Since our predictions leverage

the current state of actors at runtime, any changes caused by actors outside a particular pair are captured and utilized to make new predictions. In this way, we capture the interactions between all humans in the environment, enabling us to extend the aforementioned composition methods to capture human-human interactions for any number of actors at runtime.

It is possible, however, to train with many more humans and still encode human-human interactions, but this would greatly expand the training time since each attention combination must be tested and moreover, training with too many actors may not capture simpler interactions when only one or two actors are present.

Then, the outcome of the trained network is a binary classification $a_i, a_j \in [0, 1]$ about which actor(s), if any, in a pair the robot should pay attention to when planning its motion. The optimal label is determined by comparing the ground truth deviations caused by the robot in all attention combinations. The label used in the training for any pair of actors is one that minimizes both the deviation caused by the robot and the number of actors the robot paid attention to while causing minimal deviations.

The predictors used to obtain these labels in the training and at runtime consist of two parts: human states and robot state. To build the predictors for human states, we follow a robot-centric parametrization, in which we assume the robot is at the origin and considers all human state information to be relative to the position of the robot. Then, the parametrized states of the humans after this transformation is:

$$\boldsymbol{s}_i = [\ d_{i_x} \quad d_{i_y} \quad v_{i_x} \quad v_{i_y}\ ] \tag{5.15}$$

where $d_{i_x} = x_r - x_i$ is the relative x position of the human and $d_{i_y} = y_r - y_i$ is the relative y position of the human in the robot frame. The x and y components of each human i are also collected as part of the attributes used in predictions. We assume the robot is equipped with sensors capable of detecting these states of the actors, since the primary focus of this work is on motion planning and it has been shown that such information can be measured using cameras or LIDAR sensors [63].

Since robot low level controls are determined by the MPC, which is designed to minimize error to a goal, the only robot states we model as part of the predictor include the error between current

and desired velocity and direction/heading of the robot:

$$\boldsymbol{s}_r = [v_e, \theta_e] \tag{5.16}$$

with $v_e = v_{des} - v_r$ and $\theta_e = \theta_{des} - \theta_r$. An illustration of the training dataset used in our simulations is shown in Fig. 5.4. The robot has 7 different paths all beginning at $x = 0$, indicated by the blue lines. We train each of these robot paths with every unique pair of actor initial positions, shown by the 20 red markers, for each of the 8 directions shown on the right. This training set provides a simple proof-of-concept example, in which we are assuming that we can capture similar bounded cases at runtime, but it should be noted that with a richer and more diverse training set, the performance of the prediction would improve.

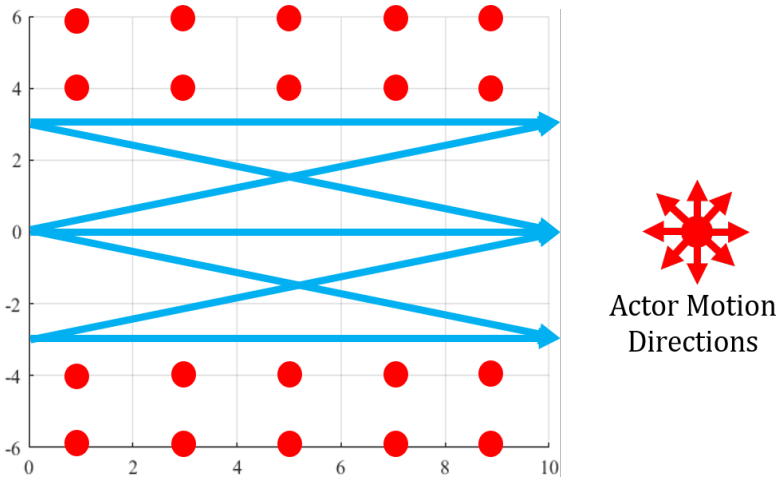


Figure 5.4: Training robot paths and actor initial positions and directions.

The actors in this training data move at a nominal speed of 1m/s in their respective directions, as indicated by studies on average human walking speed [13], but it should be noted that velocities will deviate as humans interact with other humans and the robot. The robot, throughout its operation, observes pairs of humans in multiple different configurations, which contributes to the overall quality of the training dataset. A validation set is also built and tested to ensure that the trained network will generalize well to new observations. The validation begins with similar premises as training set, but variation is introduced by adding the following Gaussian noise: $\mathcal{N}(0, 2)$ to the initial positions and the following: $\mathcal{N}(0, 0.5)$ to the velocities.

5.4.2 Attention Prediction

In this section, we introduce our attention-aware prediction model that can identify the set of most important actors $H_a \subseteq H$ in a given scenario. Our proposed predictive model consists of two modules:

- **Pairwise Interaction Module**: takes into account pairs of humans among all actors $i \in H$ to model human-human interactions in the presence of the robot.
- **Composition Module**: aggregates the individual pairwise interactions to make final binary attention predictions.

The overall attention prediction network architecture is depicted in Fig. 5.5. In the following sections, we will discuss the formulations for each part of the architecture.

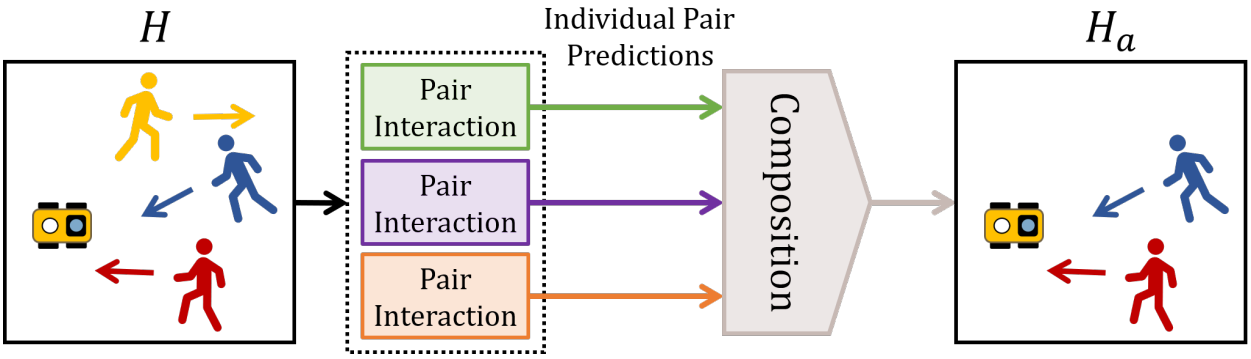


Figure 5.5: Attentive prediction network architecture

Pairwise Interaction Prediction Module

The pairwise interaction module is a deep neural network (DNN) that encodes the human-human interactions that occur between pairs of humans in the environment. The first part of the interaction module involves finding and combining the state predictor information for all unique pairs of actors. In an environment with n actors, the total number of pairs can be represented by the binomial coefficient $\binom{n}{k}$ as follows:

$$n_p = \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2} \tag{5.17}$$

For example, if the robot senses 2 actors $n_p = 1$, for 3 actors $n_p = 3$, for 5 actors $n_p = 10$, and for 10 actors $n_p = 45$. Note that $k = 2$ is fixed in our approach since we make predictions only about

pairwise interactions. The predictors for each pair can be expressed as follows:

$$p_{ij} = [\boldsymbol{s}_i,\ \boldsymbol{s}_j],\ i \neq j,\ \forall i, j \in H \tag{5.18}$$

These predictors, along with the robot information contained in α_r are then passed into the interaction module, which consists of a multi-layer perceptron (MLP) that is used to predict the attention outputs within each pair:

$$\boldsymbol{c}_{ij} = \psi(p_{ij}, \boldsymbol{s}_r; W) \tag{5.19}$$

where $\psi(\cdot)$ is a fully connected layer with ReLU activations [85] and W contains the network weights. The output of our network, $\boldsymbol{c}_{ij} \in \mathbb{R}^2$, contains attention prediction information for each human c_i and c_j. This procedure is repeated for all n_p pairs, resulting in attention estimates for each human in each pair. Then the next step becomes composing these estimates together in a way that captures whether each actor requires attention or not. The desired outcome for attention predictions is a binary variable $a_i \in [0, 1]$, where a prediction of 0 indicates the MPC will neglect to include the actor as a constraint, and vice versa. The proposed MLP network only provides this information about individual pairs of actors, bringing about to need to combine the predictions to correctly identify the appropriate actors in a given scenario. In Fig. 5.6, we demonstrate the inputs and basic architecture of the network used in this work, which consists two fully connected layers that predict two outputs, one for each actor in the pair.

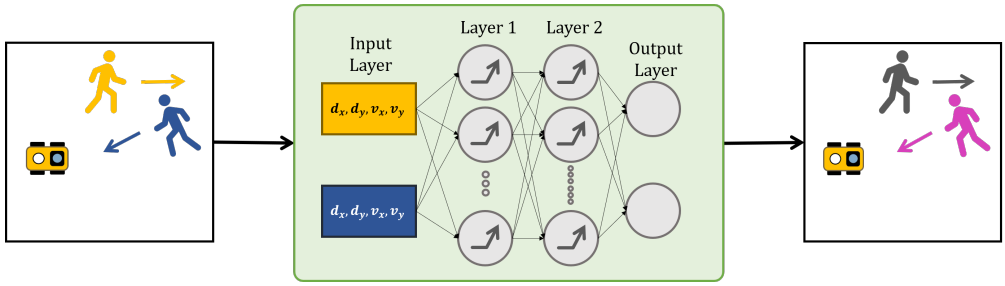


Figure 5.6: Diagram of the multi-layer perceptron used in this work to estimate attention predictions in pairwise interactions. In this diagram, the two layers have a different number of hidden units indicated by the dots in between each node. The output indicates which actors, if any, need to be modeled in the MPC, indicated by the magenta and black color of the actors after passing through the network.

In Fig. 5.7, we show a visual example of all pairs being decomposed and re-composed in a

scenario with 3 actors. Each pair is passed through the aforementioned MLP network, and the predictions are then combined in the composition module, which we describe in the next section.

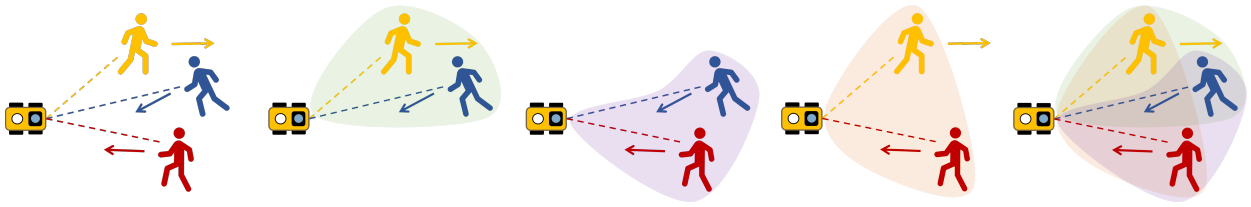


Figure 5.7: Visual example of pair decomposition and re-composition in the proposed approach

Composition Module

A number of techniques can be used to compose the previously obtained attention estimates together. Pooling techniques [15] are used often in deep learning with images to reduce dimensionality of the input and aggregate only the useful or necessary parts of a particular input. Two common pooling techniques include max-pooling and mean-pooling. Max-pooling takes into the account the maximum value within a given set of features, and utilizes this to make further predictions, while mean-pooling takes the average of all values.

While in this work, we do not use images, we take inspiration from pooling methods and model the attention estimates from each pair of actors as a set of features from which the final attention prediction must be drawn. We provide two different methods by which the final prediction can be made. First, a more conservative composition method, akin to max-pooling, is proposed. In this case, the robot pays attention to any human i if at least one pairwise prediction suggested that actor required attention:

$$a_i = \begin{cases} 1, & \max \boldsymbol{c}_i = 1 \\ 0, & \max \boldsymbol{c}_i = 0 \end{cases} \tag{5.20}$$

where $\boldsymbol{c}_i$ is the set of all attention estimates for an actor i: $c_{ij}\ \forall j \in H,\ i \neq j$. This approach provides conservative estimates since it only considers the maximum value within $\boldsymbol{c}_i$, ignoring all other pairs where the actor may not have required attention.

Our less conservative approach for composition draws inspiration from mean pooling, in which the attention prediction is sensitive to how often attention was required for an actor i in pairwise

decomposition. The attention prediction using this method can be computed as follows:

$$a_i = \begin{cases} 1, & \frac{1}{n-1}\sum \boldsymbol{c}_i \geq \beta \\ 0, & \frac{1}{n-1}\sum \boldsymbol{c}_i < \beta \end{cases} \tag{5.21}$$

where β is a threshold for predictions, typically set to $\beta = 0.50$, and $n-1$ is the number of attention estimates for each actor. With this approach, the attention predictions are more generalized to all the pairs, but are less sensitive to the individual pairs, and can remove unnecessary considerations if an actor has very few positive attention predictions during the pairwise interaction module. Both of these approaches are viable depending on the application and task at hand for a mobile robot. We note that (5.21) may neglect to tend to some actors that (5.20) will capture, but at the same time the less conservative approach will have a lower computational cost and is more likely to find feasible solutions. In Fig. 5.8, we show a practical example of the difference between the two methods. Actor 3 is included under the conservative maximum composition and is neglected under the mean composition and the threshold is set to $\beta = 0.5$.

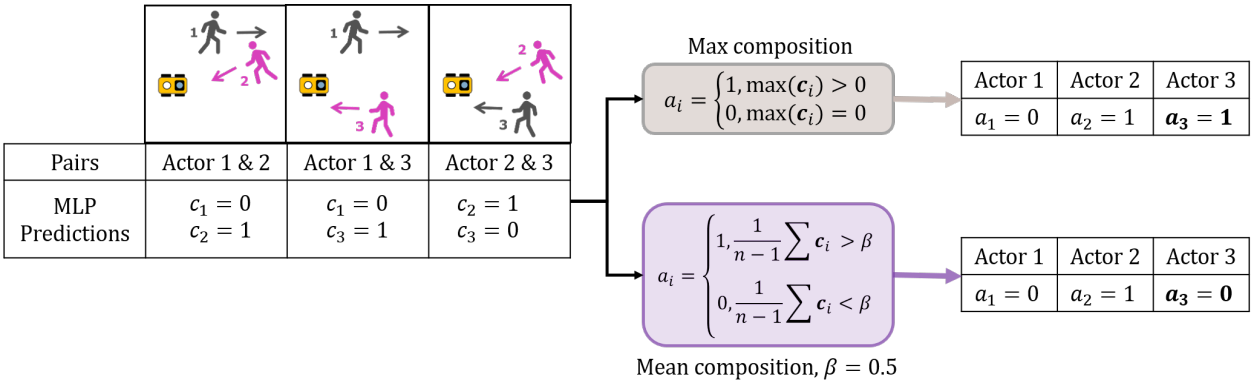


Figure 5.8: An example of the comparison between the two proposed composition methods. MLP outputs are shown on the left in a tabular form. The results of the two composition methods show that Actor 3 is captured when the maximum is taken, but neglected with the mean.

A major benefit of using the mean instead of the maximum is that actors can be given more granular attention scores in this way, which can be used in different types of motion planning algorithms, but it is difficult to reason physically about the behaviors of the robot in response to such scores [18], although our results seems to indicate that there is a correlation between these scores and the interference created, however this is left for future work.

The output of our composition method is H_a, the set of all actors the robot should pay attention to. This is then used as a part of (5.11) in the MPC optimal control problem formulation (5.8) to find inputs for the robot. A comparison of our approach under both mean and maximum methods, along with naive fully attentive and non-attentive approaches are included in the next sections.

5.5 Results

The case study investigated in this work and presented in this section consists of a robot navigating to a goal in the presence of moving humans. The robot is expected to predict which humans to pay attention to while maintaining low interference with all humans in its sensing range. We first describe the implementation details followed by the simulation results.

5.5.1 Implementation Details

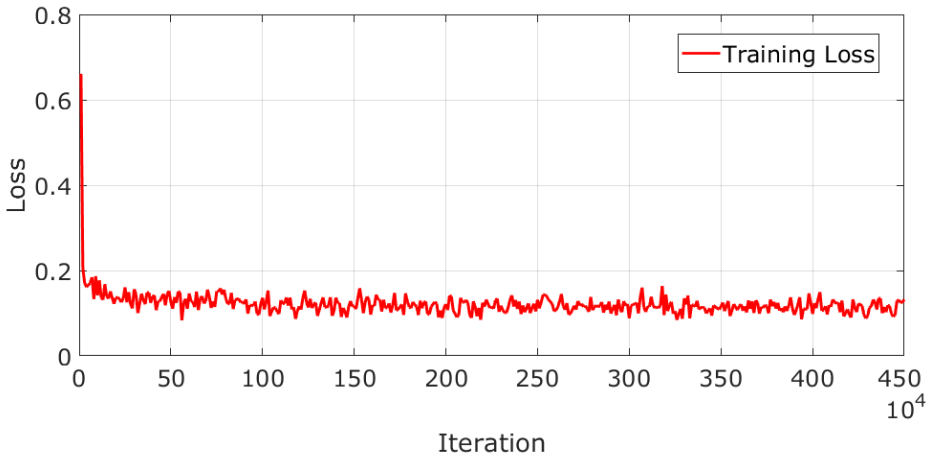


Figure 5.9: Training loss of the attention network.

We implemented the MPC using Casadi [8] in MATLAB and tested each attention combination in all trajectories shown in Fig. 5.4 to obtain labels about which actors to pay attention to in each scenario. The deep neural network was trained using the Deep Learning Toolbox integrated with Tensorflow in MATLAB over 10 epochs with a batch size of 1000 using the Adam optimizer [76] with a learning rate of 0.01 and a decay rate of 0.8 every epoch. The loss in training is measured by the standard categorical cross-entropy function [70], and the resulting network's training accuracy was 97.55% and validation accuracy was 92.30%. The training loss is plotted in Fig. 5.9.

Training the network took approximately 30 minutes and the full attention prediction at runtime consisting of both pairwise interaction and composition modules took approximately 60-90ms for up to 25 actors on a i7-1065G7 laptop CPU. The UGV simulations and experiments are run at a rate of 10Hz.

5.5.2 Simulations

We performed a series of simulations in MATLAB to test the effectiveness and viability of the proposed approach. The robot, in our simulations, traverses from its initial position to a goal at $(x, y) = (10, 0)$. We assume the actor positions and velocities can be measured by the robot at each iteration, and the actors follow the social force model described in Section 5.2.

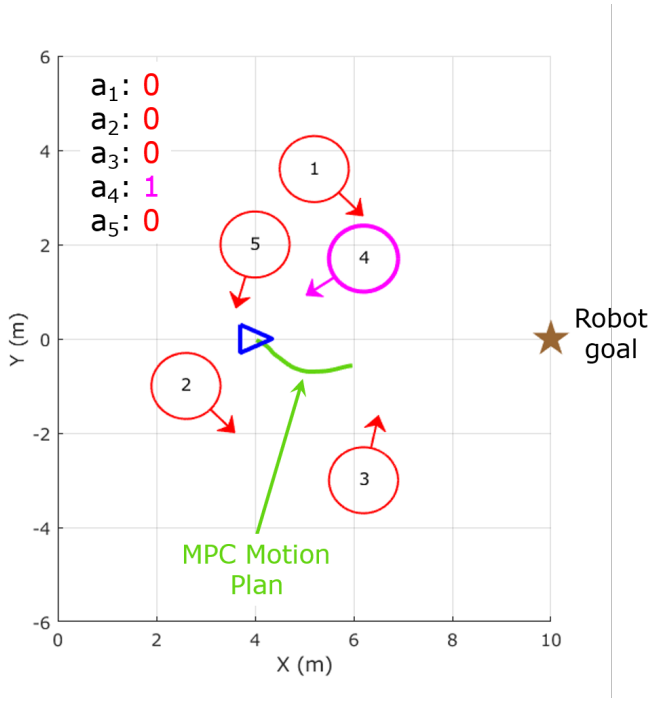


Figure 5.10: Baseline simulation of attention predictions results in a dense setting.

In the baseline simulation shown in Fig. 5.10, we show the robot at $(4, 0)$ in a dense setting surrounded by 5 actors moving in different directions. Actors in red are not given attention within the MPC framework, and the robot pays attention to the actor in purple.

In this case, the robot plans the path shown in green to avoid interfering with all actors by only paying attention to actor 4. It is worth noting when the robot does not pay attention to actor 3, a very low ($< \sigma_h$) deviation is caused by the robot's behaviors. Our approach also captures that

paying attention to actor 4 does not cause more deviation with actor 3, leaving it acceptable to ignore actor 3 in this scenario. This baseline scenario shows that our pairwise interaction approach can effectively extend to multiple actors in crowded spaces.

In the next simulation, we test a scenario with 7 actors to further examine and compare the two composition approaches presented. A sample of this scenario without attention predictions is shown in Fig. 5.11. Most of the actors in this scenario converge towards the robot's desired path while avoiding each other and the robot based on social forces.

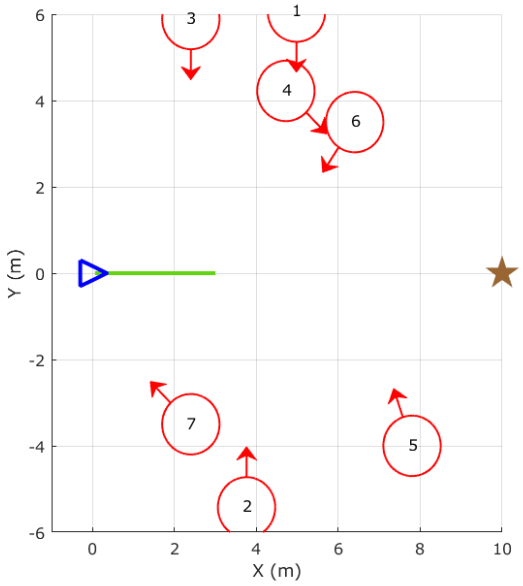
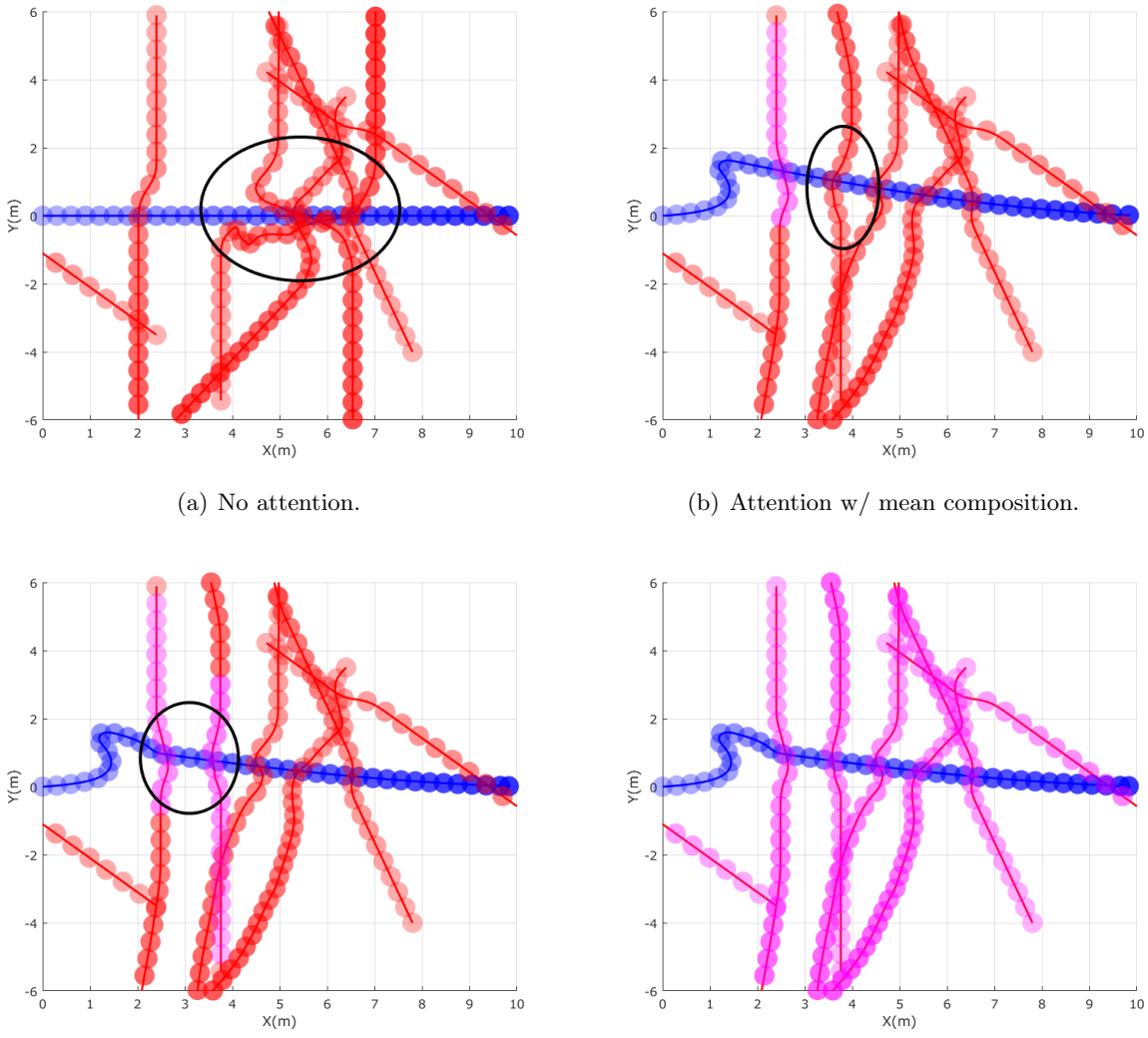


Figure 5.11: 7 actor scenario considered in this work. The actors are converging towards the robot's path and use the social force model to avoid each other and the robot.

This environment was tested in four different ways: first, we assume no attention, in which the robot moves through the environment blindly and leaves the entirety of avoidance burden on the actors. Then we test our approach under the two different composition methods we have presented, and finally, we compare the results with a full attention scenario, in which a naive MPC considers all actors in the environment equally until the robot reaches its goal. The desired effect here is that our approach is able to achieve similar results to the full attention scenario without incurring the same computational costs.

In Fig. 5.12, we show the trajectories of the different methods. The actors are in red when the robot is not paying attention and magenta when when the MPC is considering the actor as a

constraint in the optimization. It is evident here that without any attention, the robot causes major deviations in the crowd. When comparing our attention prediction approaches (Fig. 5.12(b-c)) and the full attention MPC (Fig. 5.12(d)), in which every actor is considered in the optimization, we note that the behaviors are similar and deviations caused by the robot appear minimal. The minor difference between the two composition approaches is indicated by the black circle in Fig. 5.12(b), where the actor deviates more than the same actor in Fig. 5.12(c).



(a) No attention.

(b) Attention w/ mean composition.

(c) Attention w/ maximum composition.

(d) Naive full attention.

Figure 5.12: Examples of trajectories under different methods. Our approaches perform similarly to the full attention approach, while no attention causes major deviations.

In Fig. 5.13, we show the corresponding maximum deviations for each actor in this simulation

under each of the aforementioned methods. It is clear that the no-attention method causes higher

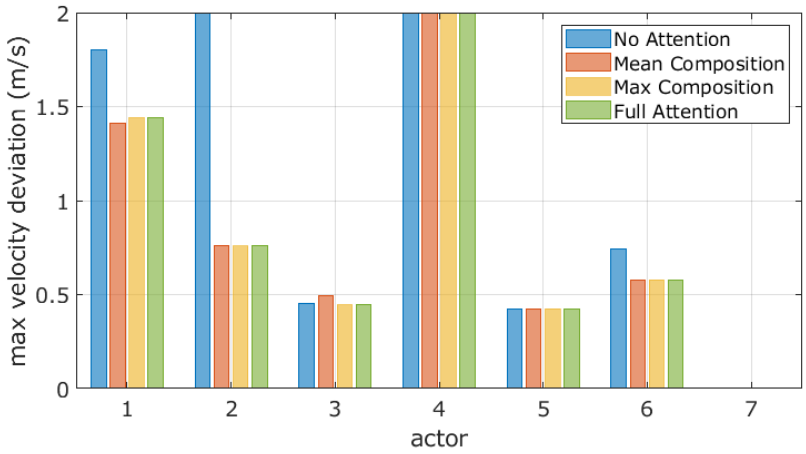


Figure 5.13: Maximum recorded actor deviations in the 7 actor simulation. It should be noted that deviations can occur on account of both robot and other actor behaviors.

deviations with actors 1, 2, and 6. Our maximum composition approach performs similarly to our mean composition approach, but a notable comparison here is that of actor 3, where our maximum composition performs more closely to the full attention method, while the mean composition approach causes more deviation. In some cases, deviations caused by the robot are difficult to estimate. Take, for example, actors 4 and 5: since both no-attention and full attention methods perform the same, we can deduce that the deviation that occurs is caused by other actors, rather than the robot. In addition to deviations, we also consider computation costs in this simulation to further validate that our attention prediction eases the computational burden on the robot. Results presented in Fig. 5.14 show average computation time for a single iteration, which includes attention prediction and MPC planning. Full attention takes the most time to complete an iteration of MPC planning, since it is including every actor as a constraint, and in a particularly dense environment, finding a solution in a limited feasible region can take longer. This further reinforces the need for our attention-aware approach, which is faster even with the prediction module enabled. The no-attention approach is the fastest since it performs no predictions and does not include the actors as constraints in the MPC.

In Fig. 5.15, we show the results over 100 tests considering between 3 and 7 actors in each scenario, much like the previous simulations shown. The y-axis in Fig. 5.15 is the proportion of the worst-case deviations for each scenario of actors. These values are obtained by collecting deviations

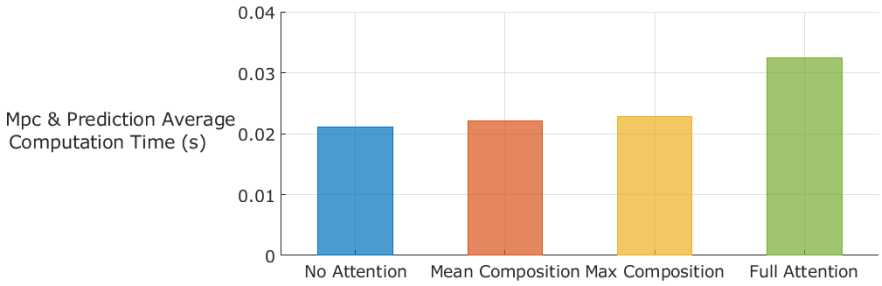


Figure 5.14: Single-iteration computation times of the methods compared in the 7 actor simulation.

under each approach and normalizing over the worst case among our tests. In this way, a high frequency at $y = 1$ indicates the robot caused large deviations, while lower values indicate the robot outperformed the worst case. We note that, while scarce, there exist cases in which full attention performs poorly. This is due to the inability of the MPC to find a solution given too restricted an environment, which can cause the robot to get stuck and results in more deviations. We also show the results of the ideal attention combination, which is determined by testing every attention combination and finding the one that minimizes the deviations. Our approach under the maximum composition method performs closest to the ideal results, while we note that the mean composition approach can cause more deviations at times, indicated by the expansion of the upper interquartile range. No attention, as expected, is typically the worst-case deviation example.

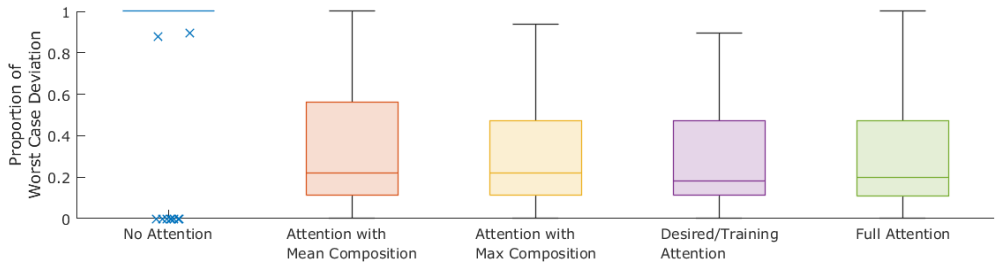
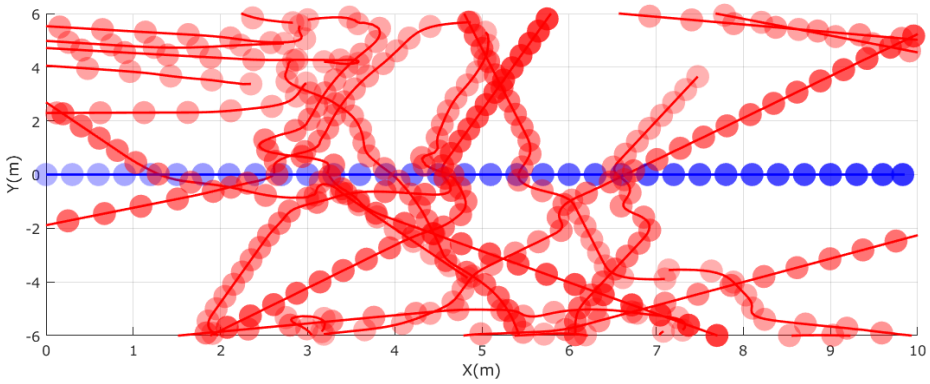


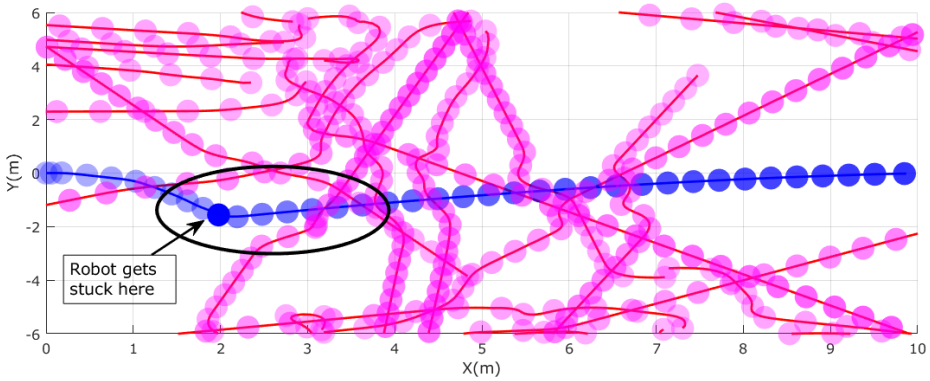
Figure 5.15: Deviation assessment over 100 tests of our attention approaches. Deviations are considered here as a proportion of worst case scenario deviations for each test.

Finally, we show a simulation in which we test our approach in a very dense scenario consisting of 24 actors. Trajectories comparing our approach against no-attention and full attention methods are shown in Fig. 5.16.

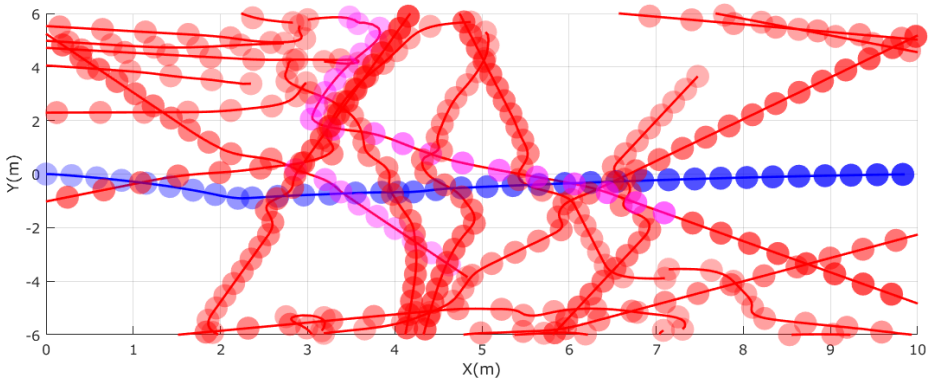
The no-attention approach in Fig. 5.16(a) places the burden on the actors and thus causes large deviations. The full attention approach shown in Fig. 5.16(b), on the other hand, considers all



(a) No attention.



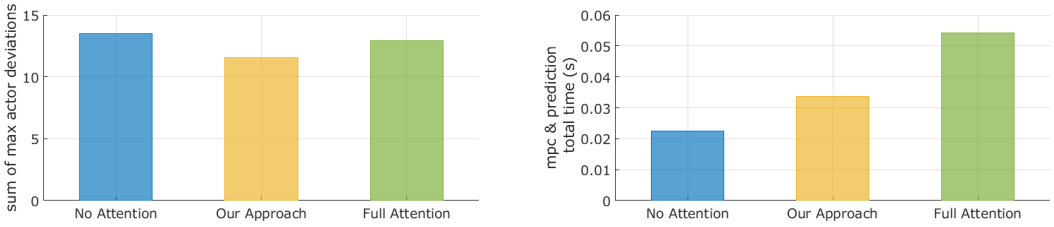
(b) Full Attention.



(c) Proposed attention-aware approach.

Figure 5.16: Examples of trajectories in a dense crowd under different methods.

actors and fails to find a solution and stops, causing deviations. Our approach, contrary to the previously presented approaches, as shown in Fig. 5.16(c) is able to find a path forward by only paying attention to two of the actors, without stopping and causing more deviations in actor paths. In Fig. 5.17, we show that our approach outperforms the no-attention and full attention approaches when comparing actor deviations and computation time.



(a) Deviation comparison.

(b) Computation time comparison.

Figure 5.17: Comparisons of computation times and deviations in dense scenario.

5.6 Discussion

In this work, we have presented a novel approach that leverages attention to promote socially acceptable and proactive robot behaviors in the presence of multiple dynamic actors. We design a neural network that considers each pair of actors to identify who the robot should pay attention to, and we follow this with a composition module that combines the pairwise predictions to assess which actors to consider given a set of actors of any size. The predictions are then fed into an MPC that considers the dynamics and corresponding future positions of actors for proactive avoiding behaviors. Our composition methods by which the network can make predictions for any number of actors demonstrate good results despite being trained just pairs of actors. This is a major benefit of our approach, as it can be time and resource consuming to train and collect data using multiple dynamic actors. Furthermore, we show that by paying attention to just a few actors, the robot can nearly replicate the behaviors as if all actors were modeled into the MPC at only a fraction of the computational cost.

5.6.1 Limitations and Future Directions

While leveraging pairwise interactions and composition can conservatively capture attention predictions, we note that it may overestimate the needed attention in some cases. However, as with any supervised learning approach, the training data must capture a good representation of what is possible for the robot to observe at runtime, and training with a larger number of actors can be resource-intensive. A possible future direction for this type of approach could be the ability to update attention predictions given new observations and context at runtime. While training on multiple actors may be difficult, considering pairs within multi-actor settings at runtime can be used to refine pairwise predictions, which would overall make motion planning better. A potential future direction can be to include an active learning-based approach through which the robot can assess and adjust predictions at runtime.

Chapter 6

Interpretable Monitoring and Recovery Under Decision Uncertainties

In this chapter, we expand our interpretable monitoring approaches beyond binary classification to multi-class problems with high decision uncertainties. Typically, learning components only provide one prediction, which may not be correct due to noise or other uncertainties in real-world settings, and only considering this prediction may lead to an unsafe situation for a mobile robot. We develop a local perturbation-based approach along with reachability analysis to identify uncertainties in predictions at runtime, to ensure safe robot motion planning. In addition, we use this chapter to further demonstrate the generality of the interpretable monitoring approaches presented thus far by extending them to detecting failures for mobile robots. Mobile robots can be impacted at runtime by external disturbances or sensing/actuation faults, all of which can cause damage to the robots themselves and create unsafe situations in the robot's environment. We demonstrate in offline training different failures the robot may experience to build at runtime an interpretable decision tree-based monitor for failure detection. We leverage MPC during this training phase to estimate deviations from desired behaviors under any of these failures, and we build a library of MPCs that takes into account different dynamics under each failure. At runtime, we not only detect and explain which failure may be affecting the system, but also use our local perturbation-based approach to consider detection uncertainties to identify the safest controller to use even when the monitor is uncertain about the cause of the failure. We note that the safest controller may not result in the highest performance for a particular failure, so we design a method by which different

controllers can be tested when safe and confidence in the highest-performing controller is reinforced with a Bayesian update. This approach is validated with both simulations and experiments of a UGV experiencing different failures at runtime. This work has been submitted and is under review at the International Conference on Robotics and Automation (ICRA) 2023.

6.1 Introduction

We find autonomous mobile robots (AMR) performing a variety of tasks that require complex decision-making, such as package delivery, search and rescue, and reconnaissance missions. Many of these robots leverage learning-based decision-making algorithms to make decisions quickly, but most of these algorithms only return one decision, which can often be incorrect due to lack of proper context during training or noise and uncertainty in observations at runtime.

In addition, due to the inherent complexity of these systems, a number of factors, such as actuator or sensor faults, can degrade the performance of the robots, which can cause critical damage to the robot itself and compromise its mission. Furthermore, these different failures can often look the same to a human observer and cause confusion for traditional methods. Learning-based approaches that deal with such problems can encode more complex interactions in measurements, and can make better decisions, but even in such critical applications, these only return one decision and do not account for uncertainties. If the robot could assess uncertainties by evaluating other decisions, particularly those that are similar to the initial decision, it might be able to take safer recovery actions.

In this chapter, we insist on this principle and investigate the failure recovery case study, where a robot must detect a system failure (e.g., on its sensors and actuators), if one is present, and account for decision uncertainties to safely correct its behavior. Specifically, we propose an uncertainty-aware and explainable decision tree (DT)-based monitor to detect at runtime which failure is affecting the system, and what other failures may be plausible given the uncertainties in the initial DT detection. Reachability analysis (RA) is then leveraged to identify safe corrective measures within a library of pre-trained Model Predictive Controllers (MPCs), which are selected in this work because of their inherent capability of predicting future states. Finally, a Bayesian performance validation scheme is proposed to reinforce (or decrease) confidence in the selected corrective measure. As

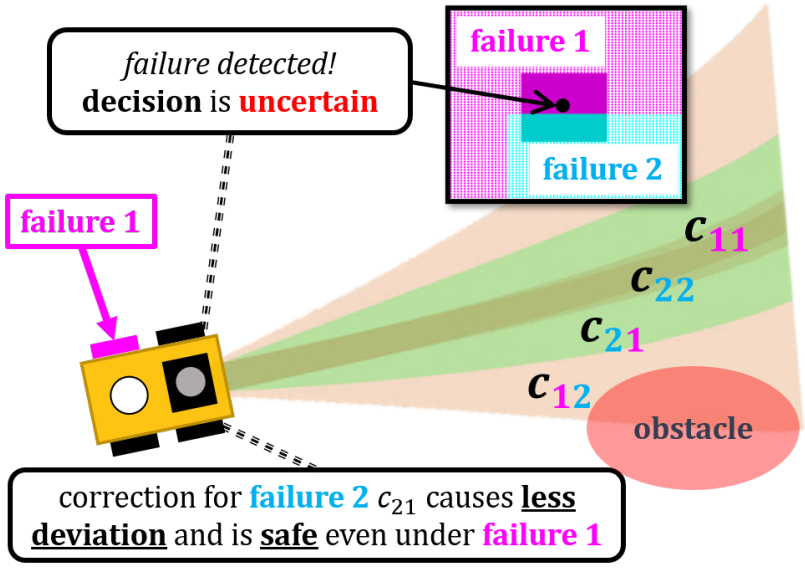


Figure 6.1: In our proposed approach, an AMR experiencing either of two failures evaluates decision uncertainties to find the safest way to correct its behaviors, even if it temporarily compromises performance.

a complementary effect, differently from other learning-enabled methods, besides quantifying the uncertainty in predictions, a human-interpretable explanation is generated which can potentially be leveraged by a human operator to further improve uncertainty assessment and validation. Shown in Fig. 6.1, is a pictorial demonstration of our approach, in which a conservative corrective measure is taken to keep a failing AMR safe in the presence of uncertainties in decision-making (i.e., the robot is uncertain if it is affected by failure 1 or 2).

This chapter presents three main contributions: 1) the design of an interpretable DT-based monitor that detects if the system is experiencing a failure and the type of failure, 2) a perturbation-based method to assess uncertainties in decision-making with a reachability-based method to find safe corrective measures given decision uncertainties and 3) a Bayesian validation scheme to increase/decrease confidence in the performance of the selected corrective measure.

6.2 Problem Formulation

Consider an autonomous mobile robot (AMR) that is navigating to a goal location. During this task, a number of different failures, such as faulty sensors or actuators (wheel encoders, propellers) or environmental disturbances (wind, ice), can cause the robot to perform poorly or fail altogether.

Distinguishing between these faults, however, can be very challenging for a human observer, standard control-based failure detection, and even some learning-based failure detection methods, if poorly trained, because of similarities between the behaviors caused by the faults, and in many cases, waiting until differences appear may not be safe. We also note that in many cases, failures have been experienced before and corrective measures that maximize performance and safety for each failure can be prepared proactively. The challenge then becomes finding a technique to detect which of these failures most closely represents the failure that might be affecting the AMR, and if the uncertainty is high, assess which of the predefined corrective measures to use to keep the system safe while collecting more data to make a more informed decision to safely recover the degraded system.

Problem 6.1 (Uncertainty Aware Failure Detection and Recovery) *Consider an autonomous robot tasked to navigate through a cluttered environment under the effect of an unknown failure f_i. Consider a set of pre-trained failures $\mathcal{F} = \{f_1, f_2, ...f_N\}$ with associated corrective measures in the form of control laws $\mathcal{C} = \{c_1, c_2, ...c_N\}$. The objective of this work is to design a framework to detect the set of possible failures $\boldsymbol{f} \subseteq \mathcal{F}$ that explain the behavior of the robot undergoing failure f_i and determine the appropriate control policy $c^* \in \mathcal{C}$ that minimizes tracking error and maximizes safety (i.e., avoids collision with surrounding obstacles):*

$$||\boldsymbol{x}(t) - \boldsymbol{x}_r(t)|| = 0, \ \ as \ t \to \infty \tag{6.1}$$

$$||\boldsymbol{x}(t) - \boldsymbol{o}_i|| > 0, \ \ \forall i = [1, \ldots, N_o] \tag{6.2}$$

where $\boldsymbol{x}(t) = [x, y]^\top$ is the position of the robot at time t, $\boldsymbol{x}_r(t)$ is the desired reference state of the robot, and $\boldsymbol{o}_i(t)$ is the position of the i^{th} obstacle and N_o is the number of obstacles.

6.3 Approach

In this section, we describe our framework for safe recovery of AMR navigation operations under degraded conditions caused by faulty actuators/sensors or environmental conditions. Our framework consists of offline and online stages, as demonstrated in Fig. 6.2.

In the offline stage, a robot performs navigation tasks using a model predictive controller (MPC) due to its model-based predictive properties, that can be leveraged to detect failures at runtime. The

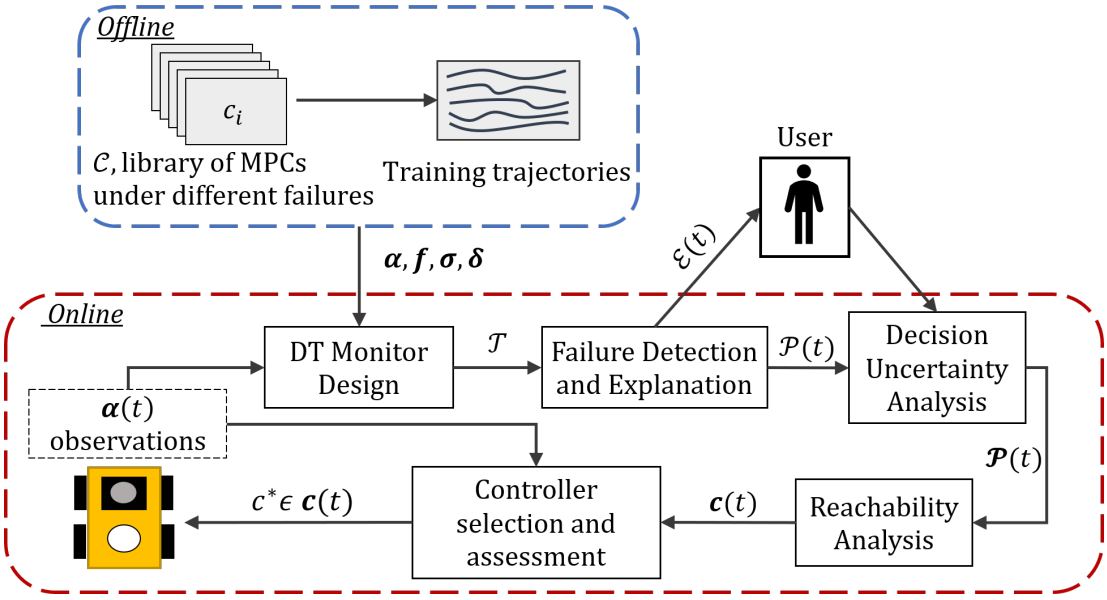


Figure 6.2: Block diagram of proposed approach.

robot is faced with different actuator and sensor failures, and a different MPC is tuned and tested to maintain a desired level of performance under each failure. This set of MPCs, $\mathcal{C}$, is used to generate the training trajectories, in which we collect observations, $\boldsymbol{\alpha}$, associated failures, $\boldsymbol{f} \in \mathcal{F}$, deviations observed for each controller-failure combination, $\sigma_{cf} \in \boldsymbol{\sigma}$, and a local perturbation distances for each observation $\boldsymbol{\delta}$, which is used at runtime to assess uncertainty in decision-making.

We build a decision tree (DT)-based monitor, $\mathcal{T}$, to detect at runtime which failure, $\mathcal{P}(t)$, might be affecting the system, and to compute an explanation, $\mathcal{E}(t)$ for this decision, which is communicated to a human user for verification. The monitor output and user input are then used for uncertainty analysis, which determines an additional set of failures, $\boldsymbol{\mathcal{P}}(t)$, that might be possible. Reachability analysis is then used to identify a set of safe corrective measures $\boldsymbol{c}(t) \subseteq \mathcal{C}$. Each corrective measure is then assessed for confidence and runtime deviations to select one that is most appropriate for the detected failure and uncertainties. This procedure is repeated, constantly re-evaluating predictions and explanations to gain confidence in the robot's decision-making and converge to safe robot behaviors under degraded conditions. In the next sections, we describe in detail each part of our approach.

6.3.1 Model Predictive Baseline Controller

A set of model predictive controllers (MPC), $\mathcal{C}$ is designed to deal with the different failures we consider in this work. Each controller $c_i \in \mathcal{C}$ is tuned appropriately based on the dynamics of each degraded system. In this work, we use MPC since it inherently provides predictions for the robot's future states $\boldsymbol{x}_p(t) = [x\ y\ \theta]^\top$, which will be compared with the observed state $\boldsymbol{x}(t)$ of the robot at runtime to facilitate failure detection and will be used for the reachability analysis performed in Sec. 6.3.3.

A key focus in this work is recovering failures that can appear very similar to a human observer, as demonstrated by the intertwined deviating trajectories in Fig. 6.3. Robots undergoing each of

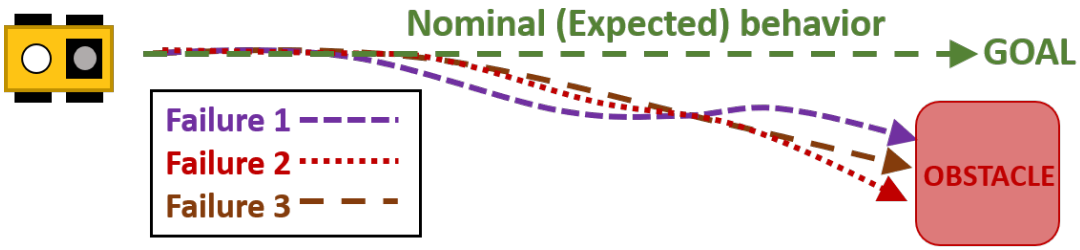


Figure 6.3: Examples of robot behaviors under different failures, showing intertwining trajectories and deviations with different colliding behaviors.

these failures may have different dynamic models, or may require different weighting parameters to achieve accurate reference tracking, and applying the incorrect controller to a misinterpreted failure may result in unsafe conditions. Thus training is performed with all combinations of failures and controllers to assess what deviations $\boldsymbol{\sigma}_{cf}$ may appear if an incorrect control policy is applied to a particular failure. Shown in Fig. 6.4 is a pictorial example of deviations that are observed when testing several controllers on a particular failure.

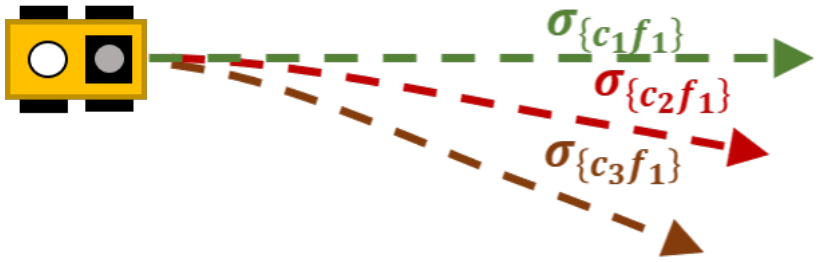


Figure 6.4: Examples of deviations obtained with different controllers on a particular failure.

6.3.2 Decision Tree Detections and Uncertainty Assessment

To detect which failure is affecting the system at runtime, we design an interpretable monitor that leverages decision trees (DT), which are a form of supervised learning that consist of interpretable white-box models [79]. DTs take in a set of input variables to make a prediction about some output. DTs are made up of a network of nodes, and the outermost nodes, known as leaves, correspond to labels given in the training (failures, in this work). In our failure detection case study, the input variables were found through experimental evaluation and are defined as follows:

$$\boldsymbol{\alpha} = [\Delta x \quad \Delta y \quad \Delta\theta \quad c_i] \tag{6.3}$$

where Δx, Δy, $\Delta\theta$ are deviations between the predicted state, $\boldsymbol{x}_p(t)$ of the MPC and the observed state, $\boldsymbol{x}(t)$ of the robot, and $c_i \in \mathcal{C}$ is the controller being deployed by the robot at the time of detection. The controller c_i is included as a categorical predictor [55], which is discrete and serves to better detect failures when any of the controllers are being used, since different deviations can be expected when different controllers are deployed under each failure.

The training process consists of testing each controller $c_i \in \mathcal{C}$ on an AMR undergoing each failure in $\mathcal{F}$ in both simulation and in hardware experiments, since testing and results are shown in both domains. The outcome of the training consists of the attributes $\boldsymbol{\alpha}$ collected at each iteration and each associated ground truth label f_i. Each pair of attributes and labels will be denoted as a sample s_i, and the collection of all samples (i.e., the entire training set) is denoted as $\mathcal{S}$. A decision tree, $\mathcal{T}$, is grown using the training data, and after taking an observation, $\boldsymbol{\alpha}(t)$, an initial failure detection can be obtained:

$$\mathcal{P}(t) = \mathcal{T}(\boldsymbol{\alpha}(t)) = f_i \in \mathcal{F} \tag{6.4}$$

After making the initial decision, a human readable explanation $\mathcal{E}(t)$ for this decision is computed by traversing the path Γ from the root of the tree, $\mathcal{V}_0$, to a prediction leaf, $\mathcal{V}_p$, taking the conjunction of each split condition, c, for the N_i nodes along the path:

$$\mathcal{E}(t) = \bigwedge_{k=1}^{N_i} c_k \quad \text{with} \quad \Gamma \mid \mathcal{P}(t) \tag{6.5}$$

Shown in Fig. 6.5 is a simple example of a DT used to make an initial detection with attributes $[x\ y] = [48\ 40]$. The failure detected is $\mathcal{P}(t) = f_0$, and through (6.5), the following explanation is obtained: $\mathcal{E}(t) = f_0$ because: $\{x < 48.5 \wedge y > 39.5\}$. The decision obtained from this procedure,

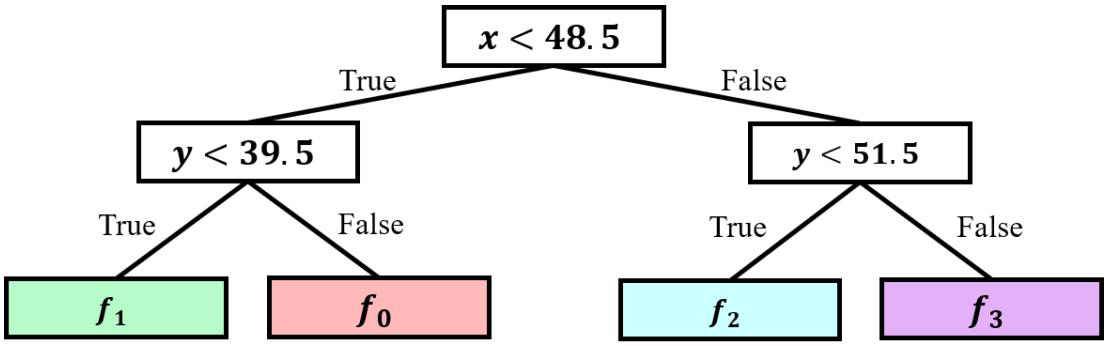


Figure 6.5: DT used for initial failure detection.

however, does not account for uncertainties, and as a result, provides information about only one outcome of the DT, which can be incorrect in the presence of noise and uncertainty. Thus, it is critical to understand what additional failures might be present by assessing these uncertainties.

Perturbation Based Uncertainty Assessment

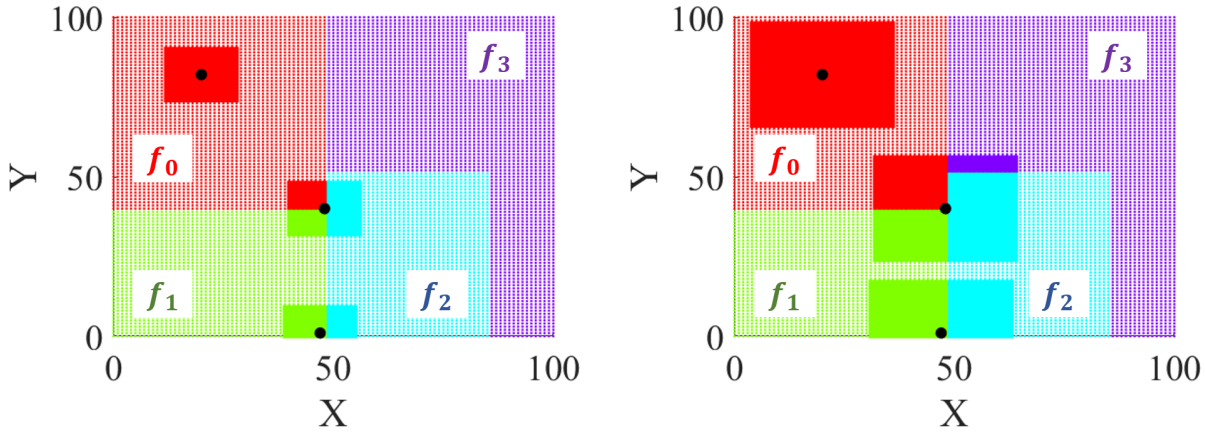
To facilitate uncertainty assessment at runtime, uncertainties are first quantified in the training by computing a local perturbation distance δ_{s_i} that characterizes the distribution of the dataset in the region around each training point s_i. In general, dense regions contain more context, resulting in accurate decisions with more certainty and vice versa. However, distance to decision boundaries plays an important role in determining uncertainty; even a decision taken in a dense region close to decision boundaries may be incorrect due to noise. To capture these uncertainties, δ_{s_i} is defined as the radius of the smallest region around training data point s_i that contains N_s observations, where N_s is a user-defined parameter that depends on the overall quality of the training data and the available computational resources [16]. The local perturbation distance, δ^*, for the runtime observation, $\boldsymbol{\alpha}(t)$, is computed by finding the corresponding value of the closest training point in $\mathcal{S}$:

$$\delta^* = \delta_{s_i},\ \text{where}\ s_i = \arg\min_{s_i} ||\boldsymbol{\alpha}(t) - s_i||\ \forall s_i \in \mathcal{S} \tag{6.6}$$

Using the computed perturbation distance, which characterizes uncertainties around $\boldsymbol{\alpha}(t)$, a perturbed dataset containing input observations and outputs (different failures), is found as follows:

$$\boldsymbol{s}(t) \subset \mathcal{S} \text{ s.t. } ||\boldsymbol{\alpha}(t) - s_i|| \leq \delta^* \ \forall s_i \in \mathcal{S} \tag{6.7}$$

Collected from within the perturbed dataset is $\boldsymbol{\mathcal{P}}(t)$, which we define as the set of all possible failures for a given set of attributes and associated uncertainties. Shown in Fig. 6.6 is an example of a uniformly distributed training dataset with two attributes, $\boldsymbol{\alpha} = [x\ y]$ and four outputs, indicated by the colored regions, akin to the DT in Fig. 6.5. The highlighted smaller rectangles inside each



(a) $\delta = 8$ (b) $\delta = 16$

Figure 6.6: Examples of local perturbations of different δ for multiple data points (black points).

figure show local perturbations with different δ around the black observations. It is evident that with larger perturbations (right), another output (f_3) is included in the local region for the data at $[x\ y] = [48\ 40]$, due to the proximity to the decision boundaries, indicating that f_3 should be included in $\boldsymbol{\mathcal{P}}$ given the observation and $\delta = 16$. It should be noted that we show this simple example to help illustrate our perturbation in a more legible way, but in this work, we have more attributes (6.3) that define decision boundaries, making visualizations more cluttered.

Shown in Fig. 6.7 is an example of a smaller local tree around the observation $[X\ Y] = [48\ 40]$ drawn from Fig. 6.6. Both trees, large and small, result in the same decision and explanation. The failure detected is $\mathcal{P}(t) = f_0$, and through (6.5), the following explanation is obtained: $\mathcal{E}(t) = f_0$ because: $\{X < 48.5 \wedge Y > 39.5\}$. The tree in Fig. 6.5, however, captures that the observation is

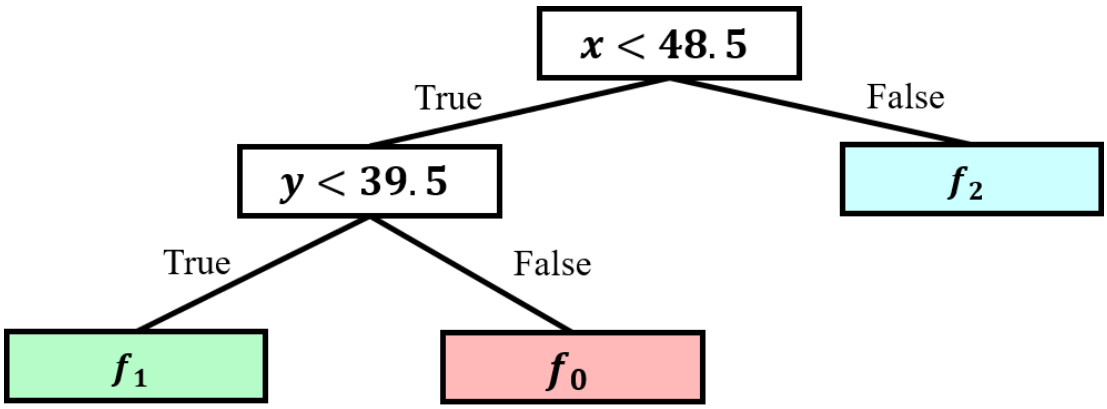


Figure 6.7: DT used for initial failure detection.

close enough to decision boundaries to also include the possibility of f_3.

Also included in this work is the ability for a human user to intervene based on the original failure detection, $\mathcal{P}(t)$,and explanation, $\mathcal{E}(t)$, to modify $\boldsymbol{\mathcal{P}}(t)$. For example, if the user decides that the system should be certain about the initial detection, then they can set $\boldsymbol{\mathcal{P}}(t) \rightarrow \mathcal{P}(t)$. The user can also add failures to $\boldsymbol{\mathcal{P}}(t)$ if their judgement suggests it is needed. In this way, the proposed approach is able to further make use of the readability and interpretability of the DT monitor towards recovering the degraded AMR.

6.3.3 Reachability Analysis and Controller Selection

To recover the AMR, we first use reachability analysis (RA) [25] to identify a set of controllers $\boldsymbol{c}(t) \subseteq \mathcal{C}$ that can be safely deployed for any of the possible failures. Reachable sets $\mathcal{R}_{cf}$ for all combinations of controllers and failures in $\boldsymbol{\mathcal{P}}(t)$ are computed by interpolating a region around the MPC predictions $\boldsymbol{x}_p(k)$ with $k = [t, t+N]$. The reachable set is bounded by maximum deviations $\boldsymbol{\sigma}_{cf}$ collected in the training. Then, the set of safe controllers, $\boldsymbol{c}(t)$ is determined by verifying that each reachable set is within an obstacle free region $\mathcal{X}(t)$:

$$\boldsymbol{c}(t) = c_i | \mathcal{R}_{c_i f_j}(t) \subseteq \mathcal{X}(t),\ \forall c_i | f_i \in \boldsymbol{\mathcal{P}}(t),\ \forall f_j \in \boldsymbol{\mathcal{P}}(t) \tag{6.8}$$

Fig. 6.8 displays an example taken from our simulations, in which each reachable set is obtained with different controllers assuming failure 1, and since $\mathcal{R}_{c_3 f_1}$ intersects with an obstacle, it is unsafe and $c_3 \notin \boldsymbol{c}(t)$. Each controller within $\boldsymbol{c}(t)$ can be safely tested, and given no other context, the controller

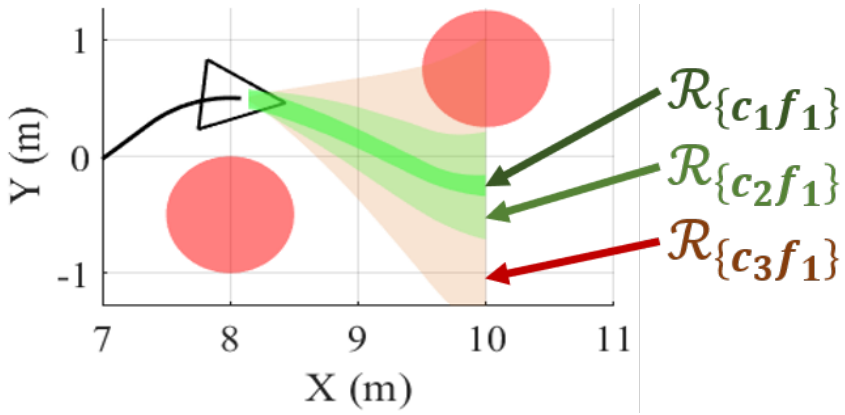


Figure 6.8: Examples of reachable sets obtained with different controllers on a particular failure.

selected, $c^* \in \boldsymbol{c}(t)$, is a conservative one with the lowest worst-case deviation for all failures in $\boldsymbol{\mathcal{P}}(t)$:

$$c^* = \min_{c}(\max_{c} \sigma_{c_i f_j}), \quad \text{for each } c_i \in \boldsymbol{c},\ \forall f_j \in \boldsymbol{\mathcal{P}} \tag{6.9}$$

Controller Confidence Assessment and Validation

Selecting a corrective measure as described above is conservative and may not necessarily maximize the performance of the degraded AMR. To improve its performance, the robot needs to gain context at runtime about the effectiveness of the selected controller and decide whether to switch to another. We model this context as a controller confidence $\Pr(c_i)$ that evolves over time based on the observed runtime deviation, which is computed as follows:

$$\eta_{c_i}(t) = ||\Delta x\ \Delta y\ \Delta\theta|| \tag{6.10}$$

where $||\cdot||$ represents the L2-norm. If using controller c_i causes less deviation than a pre-defined desired level of performance, $\eta_{c_i} < \eta^*$, the confidence in c_i should be reinforced, and vice versa. This controller confidence is formulated as an unknown discrete probability mass function (PMF) over the different controllers with uniform initial confidence estimates: $\Pr(c_i) = 1/N_c,\ \forall c_i \in \mathcal{C}$, where N_c is the total number of controllers in $\mathcal{C}$. To reflect the desired effect of growing confidence with positive reinforcement and vice versa, $\Pr(c_i)$ is updated using recursive Bayesian inference [17], given as follows:

$$\Pr(c_i|\rho_{c_i}) = \frac{\Pr(\rho_{c_i}|c_i)\Pr(c_i)}{\beta} \tag{6.11}$$

where ρ represents whether the performance criterion $\eta_{c_i} < \eta^*$ is met, and β is a normalization constant that is used to ensure that the integral of the discrete PMF is 1. A confidence threshold, γ, is set to determine and select the controller that performs best for a given failure,

$$c^* = \ c_i | \Pr(c_i) \geq \gamma \tag{6.12}$$

In this work, we set $\gamma = 0.75$, but this choice depends on the application; higher values will be more conservative and vice versa. When a controller meets this threshold, it is automatically deployed given that it is in $\boldsymbol{c}(t)$, regardless of the output of the DT monitor. When a controller is deployed and reinforced negatively, it is penalized according to (6.11), and if all controllers cause deviations $\eta_{c_i} > \eta^*, \ \forall c_i \in \boldsymbol{c}(t)$, then confidence for all controllers is reinitialized in order to allow the system to test other controllers again based on which has caused the lowest deviations at runtime:

$$c^* = \arg\min_{c} \boldsymbol{\eta} \tag{6.13}$$

where $\boldsymbol{\eta}$ is the set of most recently observed deviations for each controller in $\boldsymbol{c}(t)$. This type of situation arises when an unknown failure appears at runtime and every controller creates deviations $\eta_{c_i} > \eta^* \ \forall c_i \in \boldsymbol{c}(t)$, but runtime deviations and reachable sets suggest the system can still remain safe. We show one such case in Section 6.4.1, where a system under an unknown failure switches between controllers to remain safe and maximize performance throughout its operation. The full controller selection process in this work is shown in the flowchart in Fig. 6.9.

In a situation where no safe controller exists, $\boldsymbol{c}(t) = \varnothing$ for all failures in $\boldsymbol{\mathcal{P}}(t)$, the robot should switch into an ad-hoc fail safe mode, which can vary based on the application. One possibility is to stop operations and replan the desired path based on the deviations observed at runtime.

6.4 Results

The case study investigated in this work and presented in this section consists of an AMR navigation task in the presence of different sensor and actuator failures. The robot is expected to detect the failure affecting the system, assess uncertainties around this initial decision, and recover to a safe mode of operation.

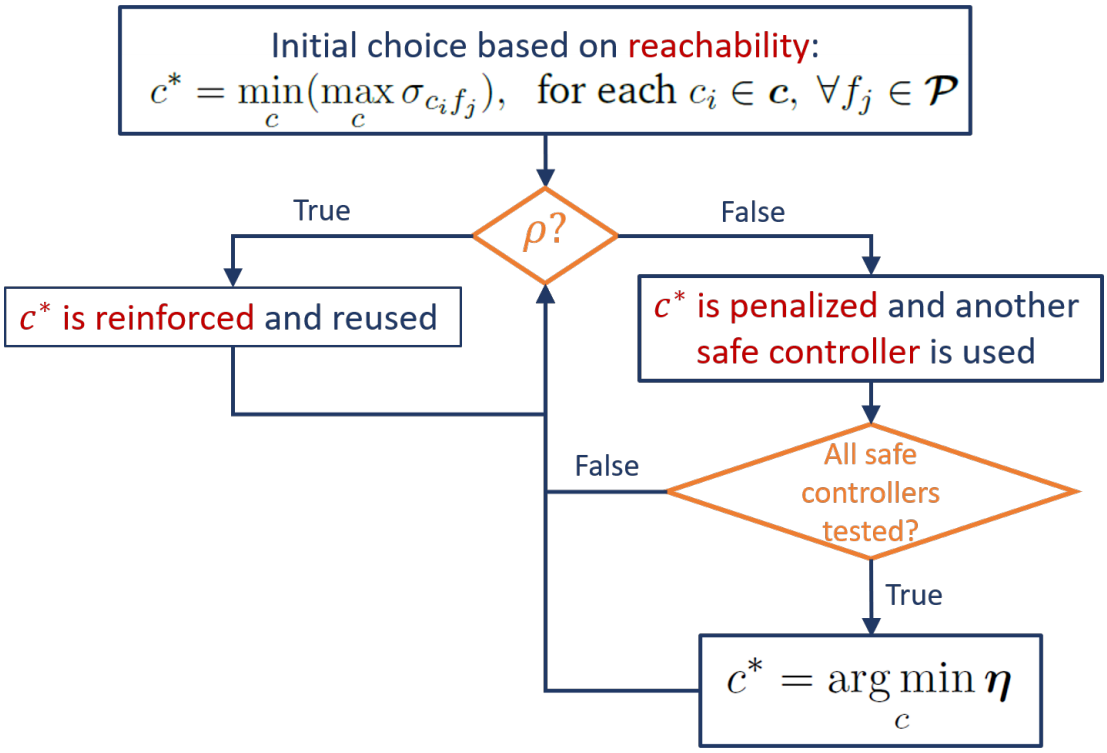


Figure 6.9: Controller selection process flowchart.

Simulations

In MATLAB simulations, the robot was tasked to move through an environment from an initial point at $(0, 0)$ to a goal at $(25, 0)$ while avoiding obstacles in the presence of failures. The simulation training set consisted of 5 failures, including: f_1: velocity based steering loss where higher speeds lead to more deviations, f_2: steering loss of 30%, f_3: icy conditions causing the robot to spin, f_4: windy conditions adding bias to the robot position, and f_5: steering loss of 10%. The MPC had a horizon of $N = 5$s, and a known map of the environment was used to define obstacle avoidance constraints for the MPC. Shown in Fig. 6.10 is a comparison between robot behavior under nominal conditions and behaviors under a failure.

The nominal trajectory (green) shows the desired behavior of the robot, while the red line shows that a robot under a failure with no correction collides with a wall. In blue, we show that under our approach, the robot is able to correct its behaviors and converge to nominal behaviors. Finally, the magenta line depicts a version of our approach in which the uncertainty assessment and controller validation are removed, and only the initial decision of the DT is utilized to select a controller. This

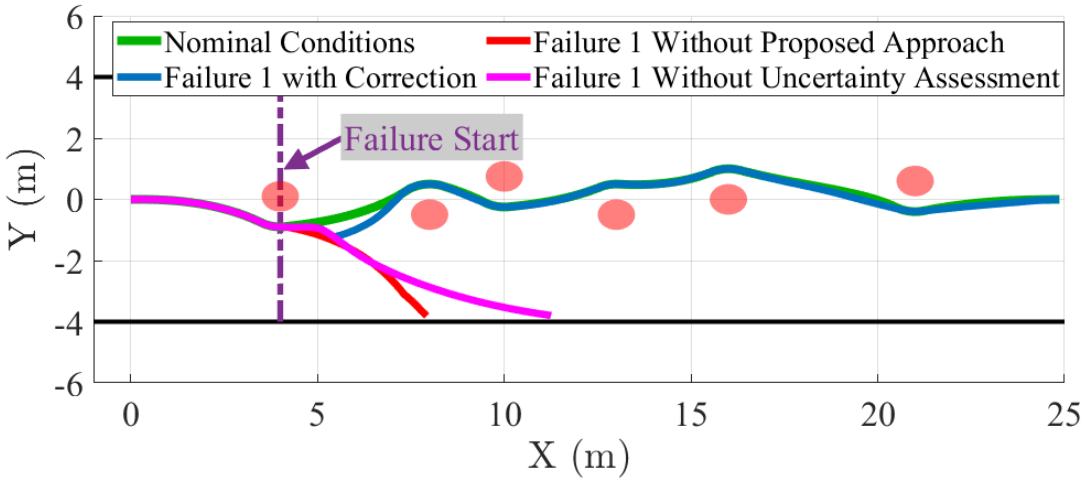
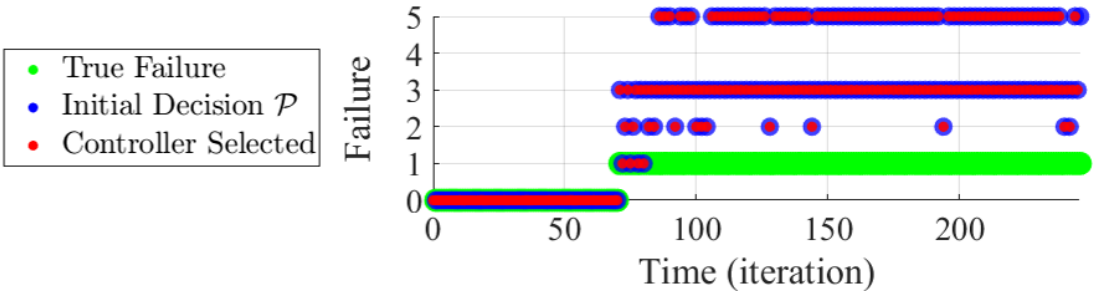


Figure 6.10: Simulation trajectories.

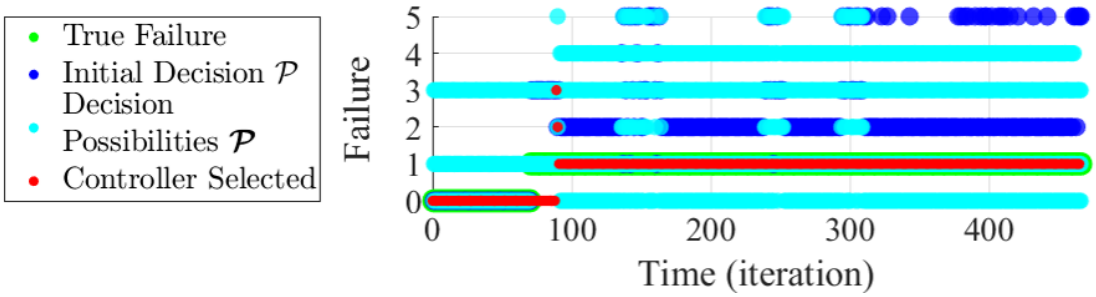
type of decision-making leads to a collision, validating that uncertainty assessment and controller validation are needed in our framework for safe robot motion.

In Fig. 6.11, we compare the decision-making results of some of the cases listed in Fig. 6.10. Fig. 6.11(a) shows the local decisions (blue markers) and controller selections (red markers) without uncertainty assessment and validation. The selected controller always matches the decision $\mathcal{P}$ in this case, and this as seen above, causes a collision. In Fig. 6.11(b), the cyan markers show the results of the uncertainty assessment $\boldsymbol{\mathcal{P}}$, and despite the poor initial decisions, our approach captures the correct failure. The system, heeding uncertainties, tests controllers c_2 and c_3 when they are safe and learns, through negative reinforcement based on deviation (6.11), to instead use c_1, which is reinforced positively after deviations below η^* are detected. In Fig. 6.12, we show the confidence in controllers over time, in which system correctly converges to c_1, even though the DT decisions were noisy and uncertain.

We also conducted another simulation to further validate the effectiveness of the proposed uncertainty aware approach with a failure that was not explicitly modeled in training set, but is bounded by failures f_2 and f_5. In the results shown in Fig. 6.13(a), we observe that the system does in fact select both c_2 and c_5 at different times since confidence in all controllers is at the initial value (Fig. 6.13(b)) due to repeated negative reinforcement, and the system is using runtime deviations (6.13) to decide which controller to select. Videos of these simulations and other controller-failure combinations are available in the supplemental materials.



(a) decision-making without uncertainty assessment



(b) decision-making with uncertainty assessment

Figure 6.11: Comparison of decision-making with and without uncertainty assessment and controller validation.

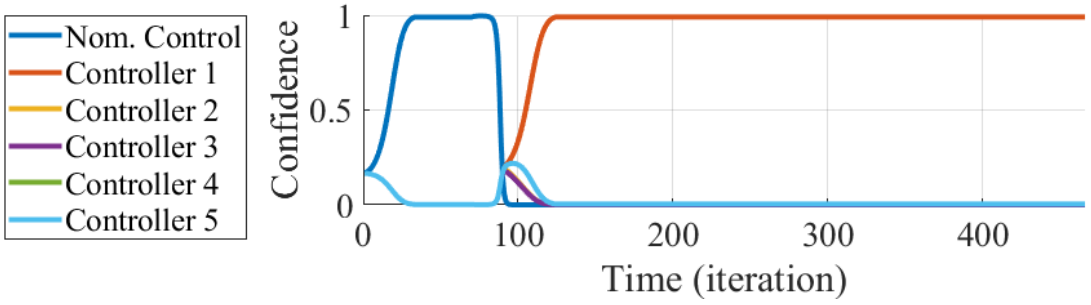
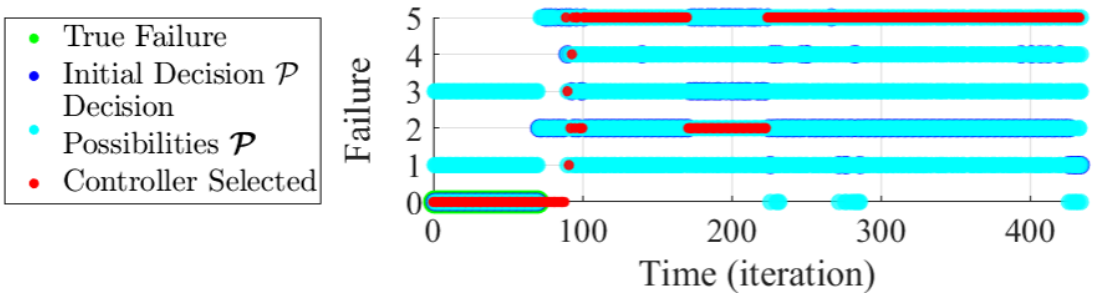
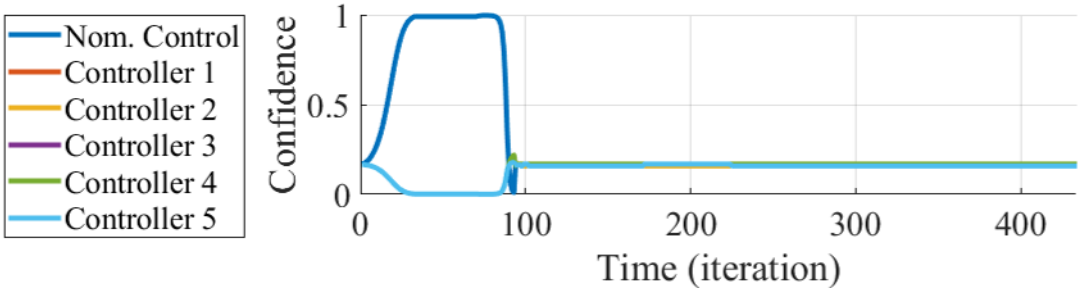


Figure 6.12: Controller validation results



(a) decision-making under unknown failure

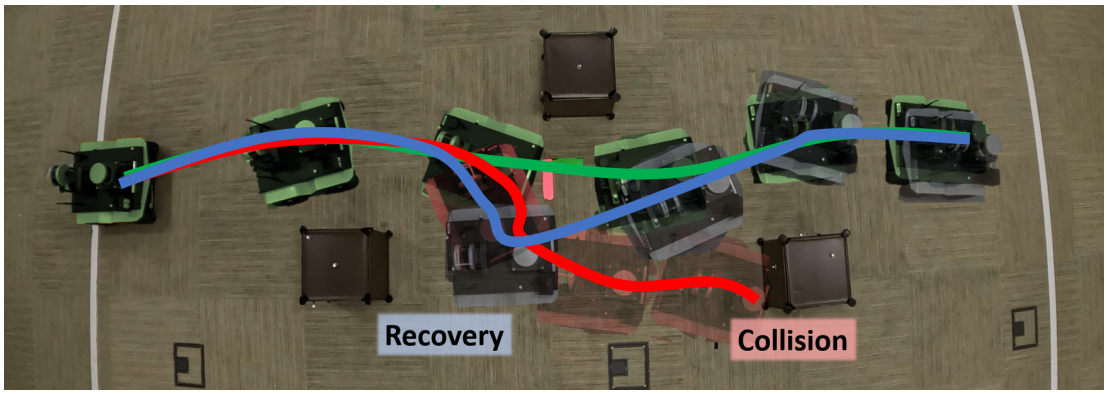


(b) Controller validation results

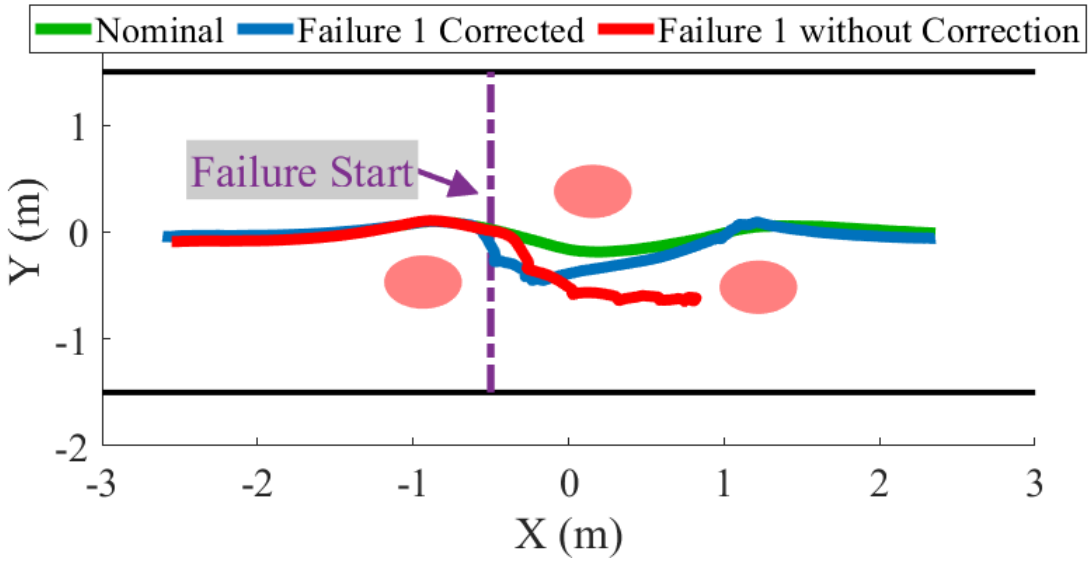
Figure 6.13: Results from simulation of unknown failure.

Hardware Experiments

Physical experiments were also conducted to test the generality and feasibility of the proposed approach. Experiment training consisted of 3 failures, which introduce different degrees (10%, 20%, and 30%) of steering loss to the vehicle. Training and testing were done on a Clearpath Jackal UGV in a lab environment with a Vicon Motion Capture system for localization. The MPC horizon was set to $N = 3$s, and our approach was executed at 10hz. The first experiment is similar to the simulation where the robot starting at $(-2.5, 0)$ has a task to safely reach a goal at $(2.5, 0)$ under a failure that is introduced at runtime. In Fig. 6.14, we show that our approach recovers the robot safely, while a robot under the same failure without our approach collides with an obstacle. In the second experiment, the robot is tasked to track an ellipse trajectory to patrol the center of the environment. Trajectories showing the effect of the proposed corrective approach are shown in Fig. 6.15, and confirm that without correction enabled, the robot diverges from its task, compromising its mission, while the proposed approach is able to maintain its performance despite the degraded conditions, and noisy failure detection.



(a) Snapshots of experiments



(b) Trajectories of experiments

Figure 6.14: Snapshots and trajectories for baseline experiments

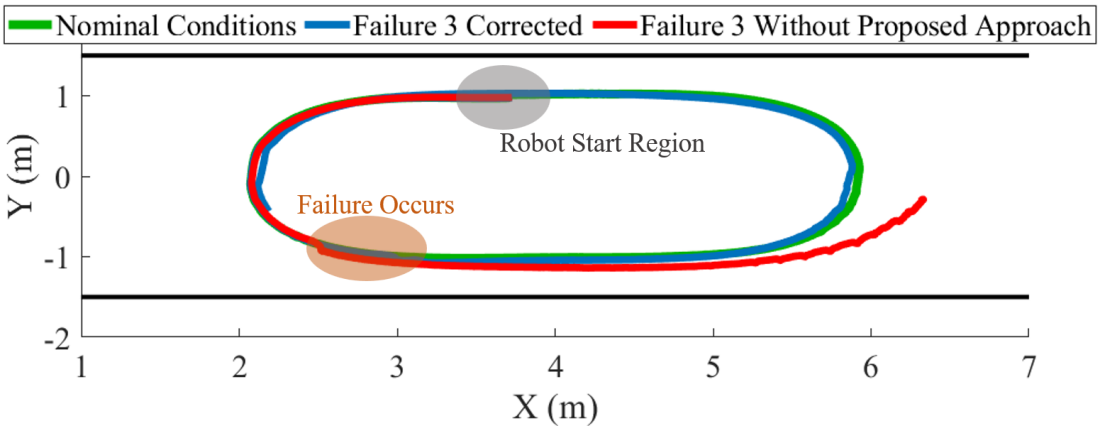
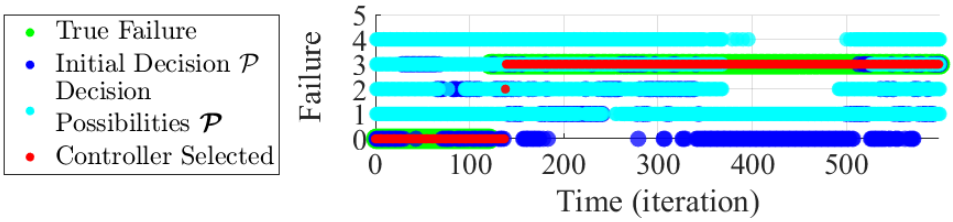
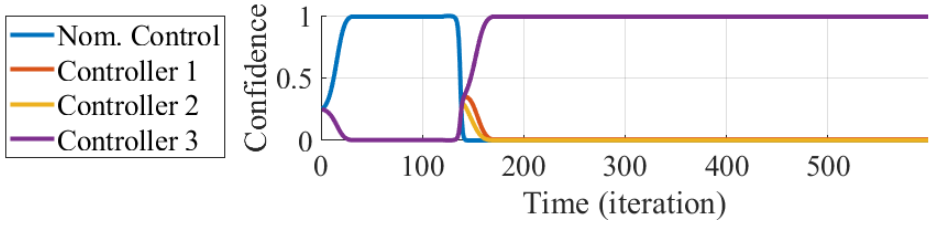


Figure 6.15: Trajectories of ellipse experiments.



(a) decision-making in ellipse trajectory



(b) Controller validation results

Figure 6.16: Results from ellipse experiment.

6.5 Discussion

In this work, we have presented a novel approach to handle uncertainties in decision-making for recovering an autonomous mobile robot from failures caused by sensor and actuator faults. We design an explainable decision tree-based monitor to detect failures and perturbation-based uncertainty assessment with a library of model predictive controllers to recover the robot to a safe mode of operation. The main benefit of our approach is that it considers and makes use of the uncertainties in the output of a learning component for robot control, promoting safe robot navigation under uncertain and noisy degraded conditions. Furthermore, the interpretability aspect of our approach empowers human users to further provide input to uncertainty analysis, adding another layer of assurance.

6.5.1 Limitations and Future Directions

As with many supervised learning approaches, the quality of the training set dictates the effectiveness and applicability of the work. While we show results in extensive simulations and experiments, dealing with unbounded, unknown, and untrained failures at runtime still remains a challenge in situations where a safe controller may not exist. However, we do note that the interpretability aspect in our approach, by including the human in the loop, can provide an avenue for runtime

updating and verification. Such issues can also be mitigated by using observed deviations to learn the degraded dynamics safely at runtime through system identification methods or reinforcement learning.

Chapter 7

Conclusions and Future Directions

In this chapter, we will conclude the dissertation with an overview of what we have accomplished and learned, followed by a discussion of real-world applications for this work and also any future directions and extensions that can build on what we have achieved thus far.

7.1 Conclusions

In this dissertation, we have focused on the problem of prediction and planning in dynamic environments under various uncertainties. We primarily focused on a social navigation case study where the source of uncertainty comes from the behaviors of nearby dynamic actors, but we also showed that the frameworks and approaches presented in this dissertation can extend to different types of uncertainties that can affect a mobile robot's performance. All techniques we have presented have been validated through extensive simulations and hardware experiments to show their applicability on real robots.

First, we presented our work on making explicit uncertainty-aware predictions about the future intended positions of humans. Different from the state-of-the-art in prediction and planning, our method leverages Hidden Markov Models (HMM) to predict not only where dynamic actors are most likely to be in the future, but also other less likely possibilities, accounting for uncertainties in the behaviors of dynamic actors. This uncertainty assessment was formulated into temporal stochastic reachability analysis, which informed an efficient and scalable virtual physics-based planner. A major benefit of this approach was that our predictive model was updated and improved at runtime

as new behaviors were observed without the need for a long training period that is typical of other learning-based approaches. Results showed the benefits of proactive uncertainty-aware planning over traditional reactive methods, and that learning at runtime improves robot behaviors. However, we also note that these explicit predictions, and particularly those aware of uncertainties were too restrictive at times for robots in highly dynamic and dense environments – the robot would at times pause since stochastic reachable sets were blocking its path.

To relax this restriction, we reduced the social navigation problem to a binary classification through the design of an interpretable monitoring technique that uses decision trees to not only predict, but also explain the causes for predictions about whether the robot will interfere with the path of a nearby human. The major benefit of using decision trees is that context can be provided for any prediction, and this provides major benefits to assurance and understanding of learning-enabled robot planning. The predictions and explanations in our presented approach are leveraged to provide counterfactuals, which provided options for how the robot could correct its behaviors. The ability to not only find a corrective action, but provide options is another impactful contribution toward the integration of robots into our world. In particular, this can apply to robotic systems with operators that can provide input and receive readable feedback about robot decision-making.

We also showed the strength of decision tree-based interpretable monitoring by adapting efficient and scalable reactive planners that suffer from local minima issues. We found that such approaches can be used in general to correct undesirable robot behaviors, while providing explanations and reasoning for why and how robot behaviors should be corrected. In addition to the interpretability aspect, another major benefit of decision trees is that they can be trained quickly online. We leveraged this aspect along with our HMM-based probability update to show that decision trees not only make predictions and provide explanations, but can also update quickly at runtime, and that predictions can improve immediately after new context is gained. This work was tested in the social navigation cases study, where extensive simulation and experiment results were promising, but showed that as the crowd size increases, a global solution that avoids interfering with all actors under the DT-based approach may not exist.

To further extend the social navigation problem to handle larger crowds, we looked towards model-based predictions for both robot and dynamic actors. We assumed actors follow a social force-based model, and used a model predictive controller (MPC) to estimate the future states of the

robot and actors. Through modeling human-human interactions, we identified that not all actors must be explicitly considered by the robot, since humans interact with each other and the presence of other humans alters how the robot responds to each one, and in many cases completely remove the need for the robot to consider certain actors. Thus, we paired a deep neural network (DNN) with the MPC to predict which actors are most important for the robot to consider. Notably, we included a method to compose together pairwise predictions at runtime so that attention-aware social planning can be achieved by training only on pairs of actors, rather than lots of randomized permutations. Results show that the robot is able to successfully minimize interfering with dense crowds, while paying attention to a smaller number of actors. Furthermore, we show that our attention prediction reduces computation time when compared to a naive MPC that considers all actors. Importantly, through our attention prediction, we achieve this reduction in computation time with minimal impact on the ability to avoid interference with all actors. We also introduced the notion of different composition methods, and showed some early results about how motion planning could be improved by considering more granular attention scores, rather than binary classifications. While it should be noted that the DNN is not inherently interpretable, by using the precise and optimal MPC that takes into account robot and human dynamics, we were able to understand better the robot's decision-making, particularly when compared to end-to-end learning approaches.

Finally, we showed how the approaches presented in this dissertation can be extended to cases in which the robot's decision-making itself is uncertain. Given that most learning components return only one prediction, which can be incorrect due to noise and uncertain environments, we designed a local perturbation-based technique to assess and quantify the uncertainties that were present. We then used a reachability-based method to correct the behaviors of a robot in a manner that is safe and sensitive to any of the possible predictions. The case study we investigated here involved predicting the type of failure a robot was experiencing, which also showed that interpretable monitoring approaches can extend beyond the social planning problem. The key result from this work is that assessing the uncertainties of predictions is critical to promote safe planning for mobile robots that are in highly uncertain environments dealing with challenging issues. To reinforce this, we showed through our simulation results that our learning component would have performed poorly and led to a collision had we not included the uncertainty assessment aspect.

7.2 Discussion and Future Directions

Throughout this dissertation, we have demonstrated that proactively handling uncertainties that affect a robot's behaviors is a vital component to the widespread use and integration of autonomous mobile robots into our society. We have shown extensive simulations with realistic robot models and experiments with real hardware in both a lab setting and a less structured indoor workspace setting. Through these results, we have shown that by assessing uncertainties and planning proactively, robots can behave in socially acceptable ways around humans, in dense crowds, and even when external disturbances and faults are causing them to fail and behave in undesirable ways. The applications studied in this work demonstrate the applicability of our frameworks to solve real-world robotic problems.

A notable key contribution of this dissertation is the interpretability aspect for robot decision-making, which enables operators or bystanders (i.e., pedestrians in the presence of an autonomous vehicle) to either receive explanations or reason about why a robot made a particular decision, as opposed to end-to-end learning approaches that use black boxes throughout prediction and planning phases. While we show that robot social navigation can be improved with our methods, we note that this type of work can extend to a number of real-world scenarios even beyond the case studies we have investigated. Recent developments in human-robot teaming to accomplish tasks have increased the importance of building trust and communication between "teammates," and our interpretable monitoring approaches, given that they efficiently return human-readable explanations for robot decisions, provide an avenue for humans to further understand robot decisions. A more concrete example is in robot-assisted surgery, in which a surgeon may be operating a robot that has perception and actuation capabilities, and with our interpretable frameworks, both surgeon and robot can communicate through explanations and counterfactuals to provide feedback about critical decisions that affect patient health. We attest that this is a form of social acceptability; not merely navigating around humans in an acceptable manner, but becoming an accepted part of our society in all facets.

We believe that this research is still at an early stage and there are important challenges before enabling autonomous mobile robots to fully integrate into our world. One particular consideration in our intention and interaction prediction work is that predictions can be improved by considering

other context, such as features of the robot's operating environment or cultural norms in the case of social planning. For example, dynamic actor intentions may be drastically different in a hostile battlefield, when compared to a busy transit station. Furthermore, beyond identifying important actors to include in our planner, there are a number of cases in which the concept of attention can be effectively leveraged, such as distinguishing between different types of drivers in an autonomous vehicle case study, or identifying relevant features a robot should consider to improve planning in a certain environment.

Another major consideration is that the approaches presented in this dissertation all utilize supervised learning, which typically requires a good training dataset. While we provided several approaches that can update and learn at runtime, we note that performance, in the end, relies on the quality of the data collected. A major challenge with collecting and leveraging data collected at runtime, however, is ensuring the robot's behaviors can be safe and closely approximate desired behaviors during that time. Future directions to address this issue could include studying more runtime learning approaches that are able to guarantee system safety while learning to improve behaviors. Safe reinforcement learning is a possible avenue for addressing the runtime learning problem, but as with many RL algorithms, it can take a large number of episodes to learn a reasonable policy, and finding a method to lower the number of episodes remains an open problem.

Overall, mobile robots have shown an immense potential to efficiently perform complex tasks and improve lives both in work and home environments. The proactive and uncertainty-aware planning techniques presented in this dissertation take these autonomous mobile robots one step closer to becoming a part of our society.

Bibliography

[1] Boussad Abci, Maan El Badaoui El Najjar, Vincent Cocquempot, and Gérald Dherbomez. "An informational approach for sensor and actuator fault diagnosis for autonomous mobile robots". In: *Journal of Intelligent & Robotic Systems* 99.2 (2020), pp. 387–406.

[2] Moloud Abdar et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges". In: *Information Fusion* 76 (2021), pp. 243–297. ISSN: 1566-2535.

[3] A. Adadi and M. Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160.

[4] Y. Akiba, S. Kaneda, and H. Almuallim. "Turning majority voting classifiers into a single decision tree". In: *Proceedings Tenth IEEE International Conference on Tools with Artificial Intelligence (Cat. No.98CH36294)*. 1998, pp. 224–230.

[5] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart. "Reciprocal collision avoidance for multiple car-like robots". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 360–366. DOI: 10.1109/ICRA.2012.6225166.

[6] Gianluca Amato, Francesca Scozzari, and Enea Zaffanella. "Efficient Constraint/Generator Removal from Double Description of Polyhedra". In: *Electronic Notes in Theoretical Computer Science* 307 (2014). Fifth International Workshop on Numerical and Symbolic Abstract Domains (NSAD), pp. 3–15. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2014.08.002.

[7] M. Amri, Y. Becis, D. Aubry, and N. Ramdani. "Indoor human/robot localization using robust multi-modal data fusion". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3456–3463. DOI: 10.1109/ICRA.2015.7139677.

[8] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[9] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. "A Comparison of Decision Tree Ensemble Creation Techniques". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.1 (2007), pp. 173–180.

[10] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. "Hamilton-Jacobi reachability: A brief overview and recent advances". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 2242–2253. DOI: 10.1109/CDC.2017.8263977.

[11] Somil Bansal, Andrea Bajcsy, Ellis Ratner, Anca D. Dragan, and Claire J. Tomlin. *A Hamilton-Jacobi Reachability-Based Framework for Predicting and Analyzing Human Motion for Safe Planning*. 2019. arXiv: 1910.13369 [cs.RO].

[12] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. *Combining Optimal Control and Learning for Visual Navigation in Novel Environments*. 2019. arXiv: 1903.02531 [cs.RO].

[13] Randolph Blake and Maggie Shiffrar. "Perception of human motion". In: *Annu. Rev. Psychol.* 58 (2007), pp. 47–73.

[14] Robert Bogue. "Growth in e-commerce boosts innovation in the warehouse robot market". In: *Industrial Robot: An International Journal* (2016).

[15] Abdelaziz Botalb, M Moinuddin, UM Al-Saggaf, and Syed SA Ali. "Contrasting convolutional neural network (CNN) with multi-layer perceptron (MLP) for big data analysis". In: *2018 International conference on intelligent and advanced system (ICIAS)*. IEEE. 2018, pp. 1–5.

[16] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[17] Manuel Castellano-Quero, Juan-Antonio Fernández-Madrigal, and Alfonso Garcia-Cerezo. "Improving Bayesian inference efficiency for sensory anomaly detection and recovery in mobile robots". In: *Expert Systems with Applications* 163 (2021), p. 113755.

[18] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. "Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 6015–6022.

[19] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P. How. "Socially aware motion planning with deep reinforcement learning". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), pp. 1343–1350.

[20] Liu Chengqing, Marcelo H Ang, Hariharan Krishnan, and Lim Ser Yong. "Virtual obstacle concept for local-minimum-recovery in potential-field based navigation". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 983–988.

[21] Boris V Cherkassky, Andrew V Goldberg, and Tomasz Radzik. "Shortest paths algorithms: Theory and experimental evaluation". In: *Mathematical programming* 73.2 (1996), pp. 129–174.

[22] Alessandro Corbetta, Jasper A Meeusen, Chung-min Lee, Roberto Benzi, and Federico Toschi. "Physics-based modeling and data representation of pairwise interactions among pedestrians". In: *Physical review E* 98.6 (2018), p. 062310.

[23] Mark Cutler and Jonathan P How. "Autonomous drifting using simulation-aided reinforcement learning". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 5442–5448.

[24] Donato Di Paola, Annalisa Milella, Grazia Cicirelli, and Arcangelo Distante. "An autonomous mobile robotic system for surveillance of indoor environments". In: *International Journal of Advanced Robotic Systems* 7.1 (2010), p. 8.

[25] Jerry Ding, Eugene Li, Haomiao Huang, and Claire J. Tomlin. "Reachability-based synthesis of feedback policies for motion planning under bounded disturbances". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2160–2165.

[26] S. Ding, X. Nie, H. Qiao, and B. Zhang. "A Fast Algorithm of Convex Hull Vertices Selection for Online Classification". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.4 (2018), pp. 792–806.

[27] Y. Dong, H. Su, J. Zhu, and B. Zhang. "Improving Interpretability of Deep Neural Networks with Semantic Information". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 975–983. DOI: 10.1109/CVPR.2017.110.

[28] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. "Path Planning with Modified a Star Algorithm for a Mobile Robot". In: *Procedia Engineering* 96 (2014). Modelling of Mechanical and Mechatronic Systems, pp. 59–69. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2014.12.098. URL: https://www.sciencedirect.com/science/article/pii/S187770581403149X.

[29] Gregory Dudek and Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[30] Stuart Eiffert, He Kong, Navid Pirmarzdashti, and Salah Sukkarieh. "Path planning in dynamic environments using Generative RNNs and Monte Carlo tree search". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10263–10269.

[31] Michael Everett, Yu Fan Chen, and Jonathan P How. "Motion planning among dynamic, decision-making agents with deep reinforcement learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3052–3059.

[32] Takahiro Ezaki, Daichi Yanagisawa, Kazumichi Ohtsuka, and Katsuhiro Nishinari. "Simulation of space acquisition process of pedestrians using Proxemic Floor Field Model". In: *Physica A: Statistical Mechanics and its Applications* 391.1 (2012), pp. 291–299. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2011.07.056. URL: https://www.sciencedirect.com/science/article/pii/S0378437111006054.

[33] Gonzalo Ferrer, Anais Garrell, and Alberto Sanfeliu. "Robot companion: A social-force based approach with human awareness-navigation in crowded environments". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1688–1694.

[34] Paolo Fiorini and Zvi Shiller. "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772.

[35] Jaime F. Fisac, Andrea Bajcsy, Sylvia L. Herbert, David Fridovich-Keil, Steven Wang, Claire J. Tomlin, and Anca D. Dragan. "Probabilistically Safe Robot Planning with Confidence-

Based Human Predictions". In: *CoRR* abs/1806.00109 (2018). arXiv: 1806.00109. URL: http://arxiv.org/abs/1806.00109.

[36] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[37] C. D. Franco and N. Bezzo. "Interpretable Run-Time Monitoring and Replanning for Safe Autonomous Systems Operations". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2427–2434.

[38] Y. Gao, P. B. Luh, H. Zhang, and T. Chen. "A modified social force model considering relative velocity of pedestrians". In: *2013 IEEE International Conference on Automation Science and Engineering (CASE)*. 2013, pp. 747–751. DOI: 10.1109/CoASE.2013.6654008.

[39] Akhil Garg and Kang Tai. "Comparison of statistical and machine learning methods in modelling of data with multicollinearity". In: *International Journal of Modelling, Identification and Control* 18.4 (2013), pp. 295–312.

[40] Shuzhi Sam Ge and Yun J Cui. "Dynamic motion planning for mobile robots using potential field method". In: *Autonomous robots* 13.3 (2002), pp. 207–222.

[41] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. "Local rule-based explanations of black box decision systems". In: *arXiv preprint arXiv:1805.10820* (2018).

[42] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. *A Survey Of Methods For Explaining Black Box Models*. 2018. arXiv: 1802.01933 [cs.CY].

[43] Pinyao Guo, Hunmin Kim, Nurali Virani, Jun Xu, Minghui Zhu, and Peng Liu. "RoboADS: Anomaly Detection Against Sensor and Actuator Misbehaviors in Mobile Robots". In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2018, pp. 574–585. DOI: 10.1109/DSN.2018.00065.

[44] M. S. M. Hashim, T. Lu, and H. H. Basri. "Dynamic obstacle avoidance approach for car-like robots in dynamic environments". In: *2012 International Symposium on Computer*

Applications and Industrial Electronics (ISCAIE). 2012, pp. 130–135. DOI: 10.1109/ISCAIE.2012.6482083.

[45] Michael Herman, Volker Fischer, Tobias Gindele, and Wolfram Burgard. "Inverse reinforcement learning of behavioral models for online-adapting navigation strategies". In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 3215–3222.

[46] E. Horváth, C. Hajdu, and P. Kőrös. "Novel Pure-Pursuit Trajectory Following Approaches and their Practical Applications". In: *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. 2019, pp. 000597–000602. DOI: 10.1109/CogInfoCom47531.2019.9089927.

[47] Zhe Huang, Aamir Hasan, and Katherine Driggs-Campbell. *Intention-aware Residual Bidirectional LSTM for Long-term Pedestrian Trajectory Prediction*. 2020. arXiv: 2007.00113 [cs.RO].

[48] Fahad Islam, Jauwairia Nasir, Usman Malik, Yasar Ayaz, and Osman Hasan. "Rrt-smart: Rapid convergence implementation of rrt towards optimal solution". In: *2012 IEEE international conference on mechatronics and automation*. IEEE. 2012, pp. 1651–1656.

[49] Dylan Jennings and Miguel Figliozzi. "Study of Sidewalk Autonomous Delivery Robots and Their Potential Impacts on Freight Efficiency and Travel". In: *Transportation Research Record* 2673.6 (2019), pp. 317–326. DOI: 10.1177/0361198119849398.

[50] Xiao-Zheng Jin, Ji-Zhou Yu, Li Zhou, and Yong-Yue Zheng. "Robust Adaptive Trajectory Tracking Control of Mobile Robots with Actuator Faults". In: *2019 Chinese Control And Decision Conference (CCDC)*. 2019, pp. 2691–2695.

[51] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. "Uncertainty-Aware Reinforcement Learning for Collision Avoidance". In: *CoRR* abs/1702.01182 (2017).

[52] Azarakhsh Keipour, Mohammadreza Mousaei, and Sebastian Scherer. "Automatic Real-time Anomaly Detection for Autonomous Aerial Vehicles". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5679–5685. DOI: 10.1109/ICRA.2019.8794286.

[53] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[54] Sujeong Kim, Stephen J. Guy, Wenxi Liu, David Wilkie, Rynson W. H. Lau, Ming C. Lin, and Dinesh Manocha. "BRVO: Predicting pedestrian trajectories using velocity-space reasoning". In: *I. J. Robotics Res.* 34 (2015), pp. 201–217.

[55] Yong Soo Kim. "Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size". In: *Expert Systems with Applications* 34.2 (2008), pp. 1227–1234.

[56] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. "Deep Reinforcement Learning for Autonomous Driving: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926.

[57] Thibault Kruse, Alexandra Kirsch, Harmish Khambhaita, and Rachid Alami. "Evaluating directional cost models in navigation". In: *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. 2014, pp. 350–357.

[58] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. "A Survey of Deep Learning Applications to Autonomous Vehicle Control". In: *IEEE Transactions on Intelligent Transportation Systems* 22.2 (2021), pp. 712–733.

[59] Steven M LaValle et al. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[60] Bertrand H Lemasson, James J Anderson, and R Andrew Goodwin. "Motion-guided attention promotes adaptive communications during social navigation". In: *Proceedings of the Royal Society B: Biological Sciences* 280.1754 (2013), p. 20122003.

[61] Lucas Liebenwein, Cenk Baykal, Igor Gilitschenski, Sertac Karaman, and Daniela Rus. "Sampling-Based Approximation Algorithms for Reachability Analysis with Provable Guarantees". In: June 2018. DOI: 10.15607/RSS.2018.XIV.014.

[62] T. X. Lin, E. Yel, and N. Bezzo. "Energy-aware Persistent Control of Heterogeneous Robotic Systems". In: *2018 Annual American Control Conference (ACC)*. 2018, pp. 2782–2787. DOI: 10.23919/ACC.2018.8431238.

[63] Timm Linder, Stefan Breuers, Bastian Leibe, and Kai O Arras. "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 5512–5519.

[64] Shuijing Liu, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3517–3524.

[65] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. "A General Framework for Uncertainty Estimation in Deep Learning". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3153–3160.

[66] Christopher W. Lynn, Lia Papadopoulos, Daniel D. Lee, and Danielle S. Bassett. "Surges of Collective Human Activity Emerge from Simple Pairwise Correlations". In: *Phys. Rev. X* 9 (1 2019), p. 011022. DOI: 10.1103/PhysRevX.9.011022. URL: https://link.aps.org/doi/10.1103/PhysRevX.9.011022.

[67] Julieta Martinez, Michael J Black, and Javier Romero. "On human motion prediction using recurrent neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2017, pp. 2891–2900.

[68] Oscar Montiel, Ulises Orozco-Rosas, and Roberto Sepúlveda. "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles". In: *Expert Systems with Applications* 42.12 (2015), pp. 5177–5191.

[69] Mehdi Moussaid and Jonathan D Nelson. "Simple heuristics and the modelling of crowd behaviours". In: *Pedestrian and Evacuation Dynamics 2012*. Springer, 2014, pp. 75–90.

[70] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[71] L. Nasraoui, L. N. Atallah, and M. Siala. "Performance study of a reduced complexity time synchronization approach for OFDM systems". In: *Third International Conference on Communications and Networking*. 2012, pp. 1–5. DOI: 10.1109/ComNet.2012.6217742.

[72] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 610–617.

[73] Ali Gurcan Ozkil, Zhun Fan, Steen Dawids, Henrik Aanes, Jens Klestrup Kristensen, and Kim Hardam Christensen. "Service robots for hospitals: A case study of transportation tasks in a hospital". In: *2009 IEEE international conference on automation and logistics*. IEEE. 2009, pp. 289–294.

[74] Luigi Palmieri, Sven Koenig, and Kai O Arras. "RRT-based nonholonomic motion planning using any-angle path biasing". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 2775–2781.

[75] Min Gyu Park, Jae Hyun Jeon, and Min Cheol Lee. "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing". In: *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*. Vol. 3. IEEE. 2001, pp. 1530–1535.

[76] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch". In: (2017).

[77] R Peddi. *UVA-AMR Human-Robot Navigation-Dataset*. `https://github.com/rahulpeddi/human-robot-navigation-datasets`.

[78] R. Peddi, C. Di Franco, S. Gao, and N. Bezzo. "A Data-driven Framework for Proactive Intention-Aware Motion Planning of a Robot in a Human Environment". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[79] Rahul Peddi and Nicola Bezzo. "An Interpretable Monitoring Framework for Virtual Physics-Based Non-Interfering Robot Social Planning". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5262–5269.

[80] Rahul Peddi and Nicola Bezzo. "Interpretable Run-Time Prediction and Planning in Co-Robotic Environments". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 2504–2510.

[81] Rahul Peddi, Carmelo Di Franco, Shijie Gao, and Nicola Bezzo. "A data-driven framework for proactive intention-aware motion planning of a robot in a human environment". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5738–5744.

[82] Praveena Penmetsa, Pezhman Sheinidashtegol, Aibek Musaev, Emmanuel Kofi Adanu, and Matthew Hudnall. "Effects of the autonomous vehicle crashes on public perception of the technology". In: *IATSS Research* 45.4 (2021), pp. 485–492. ISSN: 0386-1112. DOI: https://doi.org/10.1016/j.iatssr.2021.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0386111221000224.

[83] M. Phillips and M. Likhachev. "SIPP: Safe interval path planning for dynamic environments". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5628–5635. DOI: 10.1109/ICRA.2011.5980306.

[84] Quazi Marufur Rahman, Peter Corke, and Feras Dayoub. "Run-Time Monitoring of Machine Learning for Robotic Perception: A Survey of Emerging Trends". In: *CoRR* abs/2101.01364 (2021).

[85] Prajit Ramachandran, Barret Zoph, and Quoc V Le. "Searching for activation functions". In: *arXiv preprint arXiv:1710.05941* (2017).

[86] John H Reif and Hongyan Wang. "Social potential fields: A distributed behavioral control for autonomous robots". In: *Robotics and Autonomous Systems* 27.3 (1999), pp. 171–194.

[87] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[88] Cynthia Rudin and Joanna Radin. "Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition". In: *Harvard Data Science Review* 1.2 (2019), pp. 10–1162.

[89] Sunil Srivatsav Samsani and Mannan Saeed Muhammad. "Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5223–5230.

[90] Adarsh Jagan Sathyamoorthy, Utsav Patel, Tianrui Guan, and Dinesh Manocha. "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4352–4359.

[91] Adarsh Jagan Sathyamoorthy, Utsav Patel, Moumita Paul, Nithish K Sanjeev Kumar, Yash Savle, and Dinesh Manocha. "CoMet: Modeling group cohesion for socially compliant robot navigation in crowded scenes". In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 1008–1015.

[92] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018), pp. 187–210.

[93] S. Srungarapu, D. P. Reddy, K. Kothapalli, and P. J. Narayanan. "Fast Two Dimensional Convex Hull on the GPU". In: *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*. 2011, pp. 7–12. DOI: 10.1109/WAINA.2011.64.

[94] Jan Steinbrener, Konstantin Posch, and Jürgen Pilz. "Measuring the Uncertainty of Predictions in Deep Neural Networks with Variational Inference". In: *Sensors* 20.21 (2020). ISSN: 1424-8220.

[95] Xiaoxun Sun, William Yeoh, and Sven Koenig. "Moving target D* lite". In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. 2010, pp. 67–74.

[96] Annalisa T. Taylor, Thomas A. Berrueta, and Todd D. Murphey. "Active learning in robotics: A review of control principles". In: *Mechatronics* 77 (2021), p. 102576. ISSN: 0957-4158.

[97] P. Trautman and A. Krause. "Unfreezing the robot: Navigation in dense, interacting crowds". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 797–803. DOI: 10.1109/IROS.2010.5654369.

[98] B. Tribelhorn and Z. Dodds. "Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 1393–1399. DOI: 10.1109/ROBOT.2007.363179.

[99] J. van den Berg, Ming Lin, and D. Manocha. "Reciprocal Velocity Obstacles for real-time multi-agent navigation". In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 1928–1935. DOI: 10.1109/ROBOT.2008.4543489.

[100] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. "Reciprocal n-body collision avoidance". In: *Robotics research*. Springer, 2011, pp. 3–19.

[101] Allan Wang, Christoforos Mavrogiannis, and Aaron Steinfeld. "Group-based Motion Prediction for Navigation in Crowded Environments". In: *5th Annual Conference on Robot Learning*. 2021.

[102] Jianguo Wang, Gongxing Wu, Lei Wan, Yushan Sun, and Dapeng Jiang. "Recurrent neural network applied to fault diagnosis of Underwater Robots". In: *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*. Vol. 1. 2009, pp. 593–598. DOI: 10.1109/ICICISYS.2009.5357773.

[103] Z. Wang, P. Jensfelt, and J. Folkesson. "Multi-scale conditional transition map: Modeling spatial-temporal dynamics of human movements with local and long-term correlations". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 6244–6251. DOI: 10.1109/IROS.2015.7354268.

[104] David T Wooden. "Graph-based path planning for mobile robots". PhD thesis. Georgia Institute of Technology, 2006.

[105] Fen Xia, Wensheng Zhang, Fuxin Li, and Yanwu Yang. "Ranking with Decision Tree". In: *Knowl. Inf. Syst.* 17.3 (Dec. 2008), pp. 381–395. ISSN: 0219-1377.

[106] Yanyu Xu, Zhixin Piao, and Shenghua Gao. "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5275–5284.

[107] Esen Yel, Taylor J. Carpenter, Carmelo Di Franco, Radoslav Ivanov, Yiannis Kantaros, Insup Lee, James Weimer, and Nicola Bezzo. "Assured Runtime Monitoring and Planning: Toward Verification of Neural Networks for Safe Autonomous Operations". In: *IEEE Robotics and Automation Magazine* 27.2 (2020), pp. 102–116. DOI: 10.1109/MRA.2020.2981114.

[108] Shuyou Yu, Matthias Hirche, Yanjun Huang, Hong Chen, and Frank Allgöwer. "Model predictive control for autonomous ground vehicles: a review". In: *Autonomous Intelligent Systems* 1 (Dec. 2021).