Occlusion-Aware Navigation of Autonomous Mobile Robots in Unknown, Unstructured and Dynamic Environments

---

A

Doctoral Dissertation

Presented to
the Faculty of the School of Engineering and Applied Sciences

The University of Virginia

---

in partial fulfillment
of the requirements for the degree

Doctor of Philosophy

by

Jacob D Higgins

2024

APPROVAL SHEET

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

Doctor of Philosophy

Author: Jacob D Higgins

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

Advisor:	Nicola Bezzo
Committee Member:	Ye Sun
Committee Member:	Sebastian Elbaum
Committee Member:	Tariq Iqbal
Committee Member:	Enrico Bini

Occlusion-Aware Navigation of Autonomous Mobile Robots in Unknown, Unstructured and Dynamic Environments

by

Jacob D Higgins

M.S., Physics
The University of Virginia, 2019

B.S., Physics
University of Virginia, 2017

Abstract

The interest in autonomous mobile robots (AMR) is fast growing in the private, military, and commercial sectors for its promise to revolutionize key components of many industries, such as logistics, structural inspection and transportation. One topic that is not as well studied by academia is the problem of motion planning for AMR in occluded environments. Many practical sensor modalities for AMR are limited to line-of-sight measurements, meaning that detected obstacles (such as a static wall) may occlude the presence of other obstacles from the robot (such as a person moving in a hallway). Occlusions introduce uncertainty into the motion planning problem, as the occluded region may or may not hide obstacles to avoid. This uncertainty is multimodal and highly unstructured, making it a unique challenge towards run time navigation.

This dissertation focuses on the problem of creating an occlusion-aware motion planning policy for AMR. In particular, this work casts the occlusion-aware navigation problem as an optimization problem in which a carefully selected cost function incorporates the desired occlusion-aware behavior, in addition to the usual go-to-goal and obstacle avoidance behavior. Throughout this work, various approaches to casting this optimization problem are explored. First, a visibility-based approach is developed that seeks to approximate the area behind occlusions and actively minimize this area via a Model Predictive Controller (MPC). This results in occlusion-aware motion that requires no pre-training and little overhead to implement within a typical autonomy stack. Second, the occlusion-aware navigation problem is cast as a risk-aware motion planning problem in which occlusion-related uncertainty introduces some element of risk towards the motion of the ego robot. Emphasis is placed upon data-efficient run time learning of common risk metrics, including Value at Risk (VaR) and Conditional Value at Risk (CVaR), requiring relatively fewer datum than other machine-learning approaches. Minimizing this risk amounts to increasing visibility around occlusions, showing how occlusion-aware navigation may be an emergent behavior from risk-aware policies. Additionally, techniques for run time optimizing over these machine-learned risk functions are developed, including a trajectory generation approach based on Model Predictive Path Integral (MPPI) control theory. Lastly, this work investigates techniques for constructing these risk-sensitive policies from data collected at run time. We accomplish this using a Quantile Temporal Difference (QTD) learning algorithm to develop a risk-sensitive policy from direct experiences of the ego robot, as well as an additional approach that infers a risk-sensitive metric from observations of dynamic obstacles near occlusions. The theoretical and practical implications of these approaches are discussed, and these techniques are validated through extensive simulations and proof-of-concept experiments, both inside and outside a controlled lab setting.

"Sunlight is the best disinfectant."
– Louis Brandeis

Acknowledgements

In many ways, this dissertation is the culmination of not only the past five years I have spent in the AMR lab, but also the past eleven years I have spent at the University of Virginia. My academic journey has been interesting to say the least, and I have many individuals I would like to thank. First and foremost I want to express my deepest gratitude to Nicola Bezzo as my advisor for his mentorship and guidance throughout my time in the UVA Engineering department. I owe him a great debt for taking me on as a graduate student when I asked to join his lab in 2019, with no (!) prior robotics experience. My learning curve was steep, but I am grateful for his patience throughout the five years I have spent under his guidance. His constant energy and encouragement has taken me farther than I ever would have imagined, and I will forever be a roboticist at heart because of him.

I would like to thank my proposal committee members: Zongli Lin, Tomonari Furukawa, Sebastian Elbaum and Enrico Bini for their insightful and constructive feedback about my research last April. I would further like to thank Sebastian and Enrico for returning as dissertation committee members, as well as Tariq Iqbal and Ye Sun for joining my dissertation committee and giving me their valuable time and attention. I would like to give a special recognition towards Enrico and Alessandro Papadopoulos for our collaboration over my time as a graduate researcher. It was a pleasure working with both of you, and a joy to have briefly met you in person.

To fellow members and friends of the AMR lab – Esen, Paul, Rahul, Shijie, Phil, Lauren, Nick, Will, Patrick, Vanessa, Isabel, and Beatrice – I could not have asked for a better group people to call my friends for the past five years. Through the stresses of late nights and tough paper submissions, you have been a constant source of support, and I am grateful to have you in my life.

I also wish to thank my friends from the Physics graduate program – Sean, Marybeth, Matt, Thomas, Anna, and Miller – for allowing me to be their friend even though I abandoned you for another department. I am extremely grateful to have remained in your lives, and it has been a pleasure to watch you guys go all the way in your respective programs.

Last but not the least, I would like to thank my family: my mother Kathy, my grandmother Gladys, and my uncles Mike, Johnny, and Michael. You guys have made me into the person I am today, and my successes are because of you. I love you all.

To fund my PhD research, I acknowledge: the National Science Foundation (NSF) through grant #1816591, the Defense Advanced Research Projects Agency (DARPA) under Contract FA8750-18-C-0090, Amazon Science through their Research Award program, and CoStar under the Autonomous Building Condition Detection and Evaluation (ABCDE) grant.

Contents

List of Figures

List of Abbreviations

AMR	Autonomous Mobile Robots
AV	Autonomous Vehicle
CVaR	Conditional Value at Risk
FOV	Field of View
FRS	Forward Reachable Set
MDP	Markov Decision Process
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral
OPC	Optimal Control Problem
POMDP	Partially Observable Markov Decision Process
VaR	Value at Risk

Chapter 1

Introduction

Autonomous mobile robots (AMR) are poised to play a major role in revolutionizing many industrial sectors, having potential applications such as commercial delivery [16], warehouse logistics [55] and automated inspection [47], just to name a few. Such applications commonly require AMR to negotiate environments with line-of-sight sensors that detect and avoid obstacles within the field of view (FOV) of the robot. This presents a problem when considering obstacles that may lie behind occlusions, since the success of negotiating an obstacle can be linked to how far in advanced the obstacle is observed, giving the robot plenty of time to react. Currently, this risk is often mitigated by having the robot move slowly through the environment so that the robot has an appropriate amount of time to react. While this motion is safe, it is often simplistic and overly conservative. The aim of this work is to investigate ways of intelligently approaching occlusion-aware navigation that can plan motion without the need to act overly conservative and simplistic.

Consider the scenario shown in Fig. 1.1 in which a robot is attempting to negotiate around an open doorway within an environment. The walls that define the doorway are obstacles to avoid, but may also occluded the presence of other obstacles within the environment. The occluded region acts as a source of uncertainty, creating a risk of sub-optimal motion at best, and a risk of collision at worst. Navigation that is unaware of these considerations may take a time-optimal (orange) path that minimizes travel distance by quickly cutting the corner. A more intelligent policy may instead round the corner, taking a longer (blue) path but reducing the risks created by the uncertainty associated with the occluding corner. A general path planning policy must balance time-optimality with occlusion-aware considerations in a way that is achievable at run time and is flexible enough to account for different environments and different types of obstacles (e.g., dynamic or static).

There are many possible approaches that focus on addressing different aspects of the occlusion-aware navigation problem. For example, one approach may be to cast the occlusion-aware navigation problem as a *visibility-based* optimization problem, planning motion that maximizes the FOV area during travel and reducing the area occluded to the robot. In this way, the robot may observe

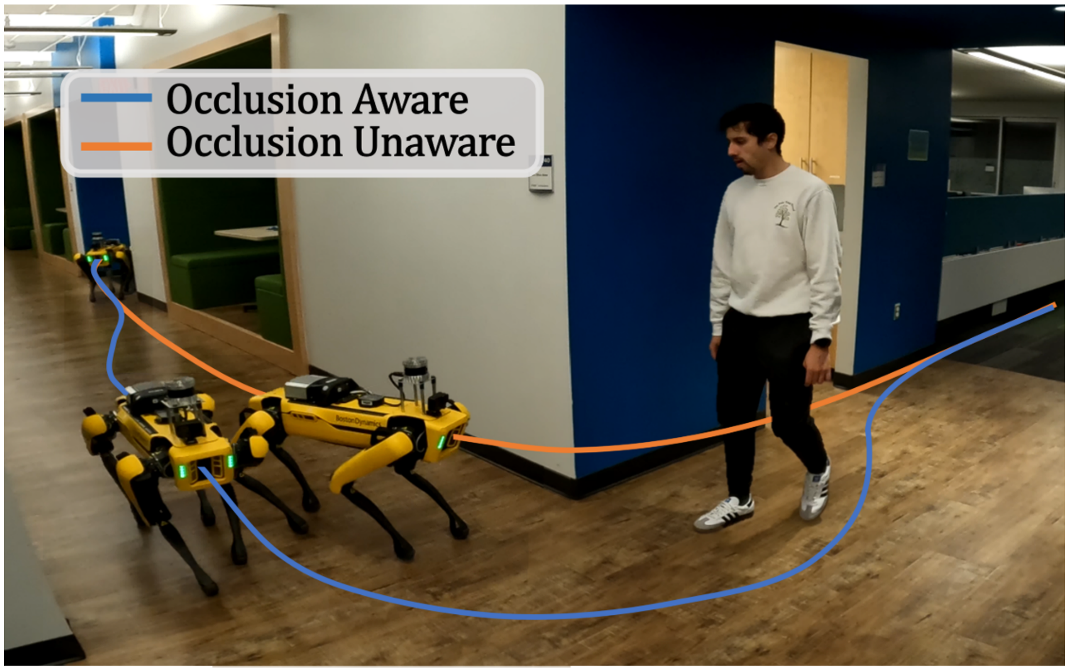


Figure 1.1: Depending on the hallway, the tracking the blue trajectory may be less risky for the robot than tracking the orange trajectory.

obstacles at a larger distance away, increasing reaction time for the onboard obstacle avoidance policy. Additionally, if dynamic obstacles are present that have their own obstacle avoidance policy, increasing reaction time may be mutually beneficial towards these dynamic obstacles as well. In Fig. 1.1, cutting the corner with the orange path would increase the occluded area behind the occluding corner, decreasing the visibility around this corner. Instead, the blue path rounds the corner at a certain distance away, decreasing the occluded area and increasing the area visible to the robot. Quantifying the amount of occluded area and using it inside the cost function of an optimization problem would thus lead to the desired occlusion-aware behavior.

A more sophisticated approach to occlusion-aware navigation recognizes that the occluded regions act a source of uncertainty for the robot, since the environment is only partially observable, and this environmental uncertainty translates into uncertainty on the quality of motion around occlusions. For example, there is a possibility that taking the orange path in Fig. 1.1 may result in smooth motion, but there is also a distinct possibility that a dynamic obstacle may emerge from this occlusion and force the ego robot into an aggressive evasive maneuver. As the occlusion creates uncertainty about what may happen, only the probability of any event can be discussed. From this perspective, the blue path may offer a better choice in that the increased distance away from the occluding corner may lessen the severity of the aggressive maneuver, lessen the probability of needing an aggressive maneuver, or both. This approach casts occlusion-aware navigation as a *risk-minimization* problem that must characterize and consider the uncertainty within the occluded regions inside the motion planning framework.

Lastly, it may be noted that throughout this work an emphasis is placed on *data-efficient* algorithms towards occlusion-aware navigation that require little to no training. This both lowers the overhead of implementing these frameworks on real-world AMR and affords a greater level of flexibility for the techniques discussed, allowing them to be applied in many different situations and platforms. This dissertation explores data-efficiency through two distinct approaches. The first is the creation of an analytical perception objective used within a visibility-based occlusion-aware framework. This analytical perception objective is defined through basic geometric properties of the surrounding occlusions, which can easily and efficiently be inferred from data available with any standard line-of-sight sensor modality. Data-efficiency is additionally explored from the perspective of risk-based approaches that attempt to *learn* a risk metric from relatively small amounts of data. For the occlusion-aware approaches present in this work, this data-efficiency allows a risk metric to be learned at run time, enabling these approaches to adjust to risks specific to the environment in which that robot is deployed.

1.1 Related Literature

In this section, an overview of related literature is detailed that provides important context for the occlusion-aware navigation techniques presented in this work. First discussed is visibility-aware navigation in which policies are created that position the robot to explicitly reduce the occluded area and promote visibility while moving. Next, general risk-aware navigation is discussed and then contextualized with the problem of occlusion-aware navigation. Finally, data-efficient learning in navigation and path planning are discussed as a means of creating occlusion-aware policies that can adjust to data collected at run time.

1.1.1 Visibility-aware Navigation

Navigation of AMR with an emphasis on visibility or perception has steadily gained attention over the past decade. Likewise, the problems addressed and the approaches developed have diverged, depending on the application and underlying system. For example, camera-based visual SLAM requires visibility of landmarks in order to reduce pose uncertainty. Thus, a number of works focus on generating trajectories that keep such landmarks within camera FOV [71, 90, 61]. Some authors use a Model Predictive Control (MPC)-based framework to promote visibility of these landmarks while commanding aggressive maneuvers [28, 49]. Additional work has focused on reducing estimation uncertainty of other obstacles in the environment, either for the purpose of improved obstacle avoidance [84] or better tracking [86]. Such ideas have also been extended to multi-robot systems that can minimize uncertainty of a single object from multiple viewpoints [82, 19]. Despite being under the umbrella of occlusion-aware motion planning, these works address a separate problem of *keeping* objects of interest within the FOV and reducing a uni-modal uncertainty of a single actor. Instead, the proposed work is concerned with using visibility to *discover* obstacles to negotiate, reducing the highly multi-modal uncertainty for general occupancy.

The discovery of obstacles in an occluded environment is not necessarily a new research topic. The watchman routing problem [22, 17, 56] is a classic computational optimization problem in which a path is planned where all unoccupied regions of a map must to observed by the line-of-sight sensor on board the robot. The robotics community has extended this concept to unknown environments, solving the watchman routing problem online as more portions of the environment are discovered [94, 25, 74, 14, 58]. The goal of these works are often mapping oriented, treating occlusions as an obstruction towards completely sensing the environment. While this is also true of the occlusion-aware navigation problem discussed in this dissertation, we consider visibility as a means of promoting better motion throughout the environment, and not necessarily to discover more of the map per se.

Research into promoting visibility as a means of improving navigation has increased in recent years, especially within the autonomous vehicle (AV) community. Authors in [5] seek to reduce occlusions due to static parked obstacles, while [4] penalizes trajectories in which occluding corners have a low angle of incidence with respect to the orientation of the AV. [36] constructs a policy to maximize information gain of these occluded regions as the AV moves near occluding intersections, and [35] extends this work by constructing a grid-based utility function that enables paths to be planned that encourages greater visibility around occlusions.

1.1.2 Uncertainty- and Risk-aware Navigation

Instead of focusing on increasing visibility around occlusions, these occlusions may be treated as a source of uncertainty for both static and dynamic obstacles. One technique to capture and mitigate this uncertainty is to model the system as a Partially Observable Markov Decision Process (POMDP) [43]. POMDP's actively understand that the current state of the system (including obstacle location) is only partially observed at any given moment and uses this partial information to plan motion. In the context of path planning for AVs, several works have attempted to characterize occluded regions as an unobservable portion of the state [13, 40, 79, 87]. These policies are often tailored specifically for AV systems and exploit the structured nature of the road network in order to apply simplifying assumptions to the problem. In the general setting, however, the POMDP has been historically intractable to solve at run time for general robotics applications, although some recent progress has been made on quickly finding approximate solutions to the POMDP [46].

Another way to address this uncertainty is to consider the "worst-case" scenario and construct a reachability set of any possible dynamic obstacle that may emerge from an occlusion. For example, authors in [69, 62] construct representations of occluded regions, predict reachable sets of potential occluded agents, and plan trajectories that avoid these reachable sets. Authors in [97] instead use bi-directional reachability to lower the computational complexity of constructing the reachable set. These reachable set approaches are often limited to AV use cases, where assumptions can be placed upon the movement of the occluded obstacles and simplify the computation of the reachable set [2, 45, 68]. Using reachability-based techniques to consider only the "worst-case" scenario can often result in overly conservative motion, which is important in safety-critical situations but leaves room for improvement for other smaller-scale systems.

Approaches that create less conservative policies may attempt to consider more of the uncertainty distribution other than just the worst-case. Chance-constrained path planning problems are one approach to this, which attempt to quantify the probability of a negative outcome happening and find a path that ensures this probability is less than a predefined amount [10, 27, 66, 15]. Chance-constraints are often non-convex, however, and generally cannot be solved at run time without overly-simplifying assumptions on the model or underlying uncertainty. Recent efforts have instead recognized that risk metrics other than the worst-case scenario may be useful within robotics [53]. Alternatives include Value at Risk (VaR) [1], which defines the quantile over a distribution, and the more popular Conditional Value at Risk (CVaR) [38, 51, 63], which defines the expectation of a distribution conditioned on being within the worst percentage of cases. Regarding occlusion-aware navigation, some works have defined custom risk metrics that help quantify the question of how "bad" are the worst possible cases, and act according to severity [77, 96].

Most of these risk-aware approaches, however, rely on a model of the underlying risk, often defined by tunable parameters that must be determined prior to deployment. In reality every environment is different, and reliable autonomy will require the robot to adapt to observations and experiences collected at run time. The next section explores this concept of learning to use data collected by the robot to make adjustments within a framework at run time in a way that efficiently leverages this data.

1.1.3 Data-efficient Learning for Navigation

Data-driven navigation allows policies to instead be learned from experience, rather than set-based rules. Reinforcement learning is a well-studied example of such a data-informed approach, with many works creating complex policies for effective planners and controllers [81, 95, 42, 41, 30]. One challenge regarding a data-informed approach to policy creation is the need for large amounts of data required to train effective policies. For example, DeepMind's DQN algorithm required millions of frames of gameplay in order to learn superhuman policies for Atari games [57], and OpenAI's Dactyl used many millions of state transitions and hours of training for human-level dexterous controls for a robotic hand [6]. While this amount of data collection is achievable through simulation, it becomes much more difficult for policies deployed on real-world systems. Within the robotics community, this leads to a concern for *data-efficient* approaches that can learn policies with relatively little data. Such approaches involve using simplified inference models that can learn with relatively little data such as an SVM [73], or, more popular, the use non-parameteric inference models such as a Gaussian Process [20, 70, 60, 21]. Most of these works, however, are concerned with learning an accurate data-informed model for the ego robot system.

With regards to navigation, a recent survey on data-informed navigation [91] has found that a majority of data-informed approaches focus on end-to-end policies, but lack proven reliability in real-world scenarios. In contrast, approaches that learn a subcomponent of a classical planner inherit safety and explainability of the classical planner while enjoying the benefits of machine learning, often requiring less data to train when compared to the complexity required for end-to-end policies [92]. In fact, a recent comparative study highlights the difficulty of end-to-end planners in generalizing to new environments in which the planner was not trained [93]. Instead, some recent work has explored learning cost maps for motion planners, such as [29] that learns a traversability cost map for a Spot quadruped from a dataset generated from a traversability oracle, or [18] that learns a cost map from a pseudo-traversability metric derived from IMU data. For social navigation, some works use inverse reinforcement learning to create a socially-aware cost function from expert-generated navigation examples [65, 54]. For these works, however, training occurs offline for large neural networks that cannot be adapted at run time. The work within this thesis instead focuses on components that are trained with relative little data, allowing training to occur online so that the robot can immediately adjust its behavior to the peculiarities of any environment in which it is deployed.

1.2 Overview of Research

The research presented in this dissertation consists of three successive parts that include: I) visibility-based occlusion-aware navigation, II) data-driven, risk-aware motion planning, and III) data-driven, risk-based occlusion-aware navigation. A final Part IV summarizes what we have gained and learned from the first three Parts. Figure 1.2 provides an overview of the research presented in this dissertation.

Beginning with **Part I**, the occlusion-aware navigation problem is cast as a visibility-based optimal path planning problem, in which the geometric area behind occlusions is estimated and actively minimized, promoting visibility around occlusions. Analytic expressions for these approx-

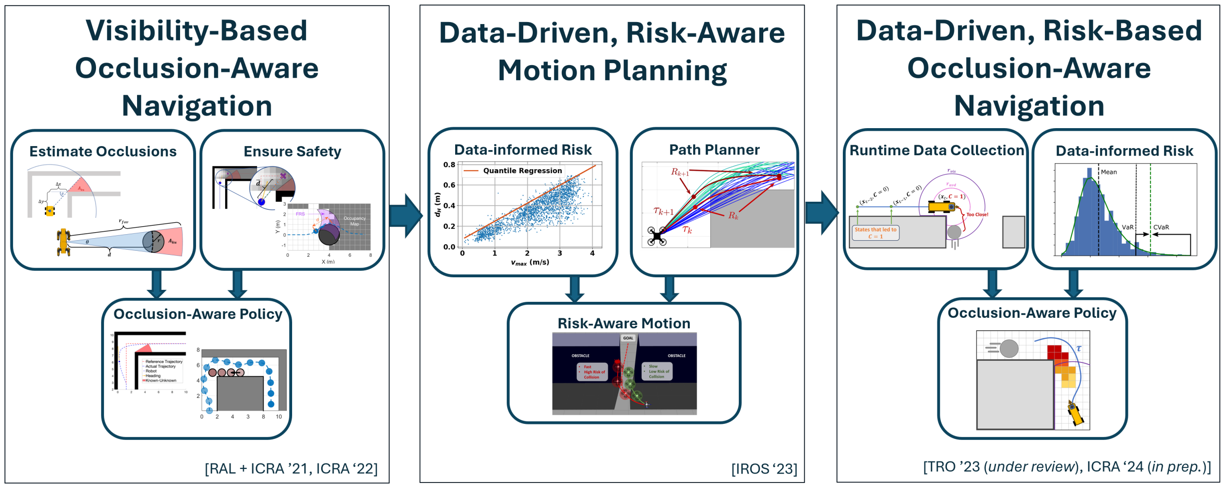


Figure 1.2: Overview of the presented research in this dissertation.

imations are constructed and motivated in this section, and are placed within the cost function of a receding horizon optimal control problem (OCP). The occlusion-aware policy is derived from solving this OCP online using numerical solvers. In **Part II**, we shift our focus towards a risk-aware navigation approach that characterizes risk of collision and plans a trajectory that actively attempts to minimize this risk. A risk metric is constructed that is used within the cost function of an OCP, so that solving this OCP results in a trajectory that minimizes this risk metric. Emphasis is also placed on constructing a data-informed risk metric that efficiently leverages data collected from the system and learns an accurate characterization of risk for different trajectories. Part II also explores runtime sampling-based approaches towards solving this OCP, which do not require the cost function to be analytically differentiable and enables a richer class of cost functions to be utilized. **Part III** applies these concepts of risk-aware navigation and data-efficient learning towards occlusion-aware path planning. We present an approach that uses data involving the ego robot's experience of a particular environment, collected at run time, to learn an occlusion-aware globally defined risk metric that is minimized over the planned path. This allows the resulting policy to actively adjust itself to avoid negative outcomes experienced by the robot at run time. Additionally, the globally defined risk metric allows the policy to also adjust to specific traffic patterns of dynamic obstacles within a particular situation, rather that treat all occlusions as the same. Before concluding Part III, we also include our preliminary work that extends beyond defining a policy using only negative outcomes experienced by the robot, and instead include an inference of what could happen if the ego robot moves close to an occlusion, given historical observations of dynamic obstacles moving around the same occlusion.

Throughout Parts I-III in this dissertation, we validate these frameworks through high-fidelity simulations and physical experiments both inside and outside the lab. Finally, in **Part IV**, we conclude the dissertation by providing insight in what we have accomplished and learned, followed by a discussion on possible directions we could take in the future.

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. As a summary for this dissertation, Part I consists of Chapter 2 and Chapter 3 which constructs an analytical approximation for geometric visibility around occlusions and actively attempts to find the optimal balance between visibility, safety (avoiding collisions) and liveliness (making it to a goal). Part II is covered by Chapter 4 and explores the data-driven risk-aware motion planning problem, and Part III is comprised of Chapter 5 and Chapter 6 which explore how these data-driven risk-based approaches may be used towards occlusion-aware navigation. Part IV concludes this dissertation by summarizing our results and discussing possible future work.

Chapter 2: Negotiating Visibility for Safe Autonomous Navigation in Occluding and Uncertain Environments

This chapter presents a novel planning framework that combines both perception and safety constraints, resulting in motion that is quick and safe when occlusions are present. Perception is satisfied using a model predictive control (MPC)-based approach to provide inputs that increase visibility around occlusions. This is achieved by engineering an analytical cost function component that approximates the geometric area behind occlusions. By including this analytical term inside the cost function of an MPC, the resulting motion policy actively seeks to improve visibility around occlusions while moving towards a goal and avoiding obstacles. Additionally, safety is promoted by modeling uncertainties as projected probabilities of occupancy derived from current observation and expected traffic motion. Improvements in visibility, safety, and speed are shown in simulations and are experimentally validated using an unmanned ground vehicle. This chapter is based on the following publication:

- Higgins, J. and Bezzo, N, “Negotiating visibility for safe autonomous navigation in occluding and uncertain environments,” *in IEEE Robotics and Automation Letters (RAL), 2021.*

Chapter 3: A Model Predictive-based Motion Planning Method for Safe and Agile Traversal of Unknown and Occluding Environments

In this chapter, the planning framework of Chapter 2 is extended to include unknown and unstructured environments. First, this extension involves the the inclusion of a new analytical term that can be used towards more generally shaped occlusions, and does not require a structured environment in order to function correctly. This new visibility cost is geometrically motivated and includes a discussion on how it may be used in unstructured environments. Second, a safety module is included that does not require a pre-constructed nominal trajectory to aid in producing occlusion-aware motion, as such a trajectory would not be available in unknown environments. This chapter is based on the following publication:

- Higgins, J. and Bezzo, N., " A model predictive-based motion planning method for safe and agile traversal of unknown and occluding environments," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9092-9098.

Chapter 4: A Model Predictive Path Integral Method for Fast, Proactive, and Uncertainty-Aware UAV Planning in Cluttered Environments

This chapter outlines an approach towards risk-aware motion planning for agile navigation that avoids the risk of collision. This is accomplished through the creation of a risk metric that can accurately establish a conservative estimate on the risk of collision for a given trajectory. This risk-metric is unique from related approaches in that it is represented as a machine-learned model, trained using data collected from a number of flight demonstrations. The number of demonstrations needed to train this component is relatively much smaller than comparative machine-learned policies, allowing a more data-efficient approach towards constructing a risk-aware policy. This risk metric is passed into the cost function of an OCP, and the resulting policy accomplishes risk-aware trajectory generation by minimizing this risk-aware component of the cost function. In order to find the solution to this OCP, a novel sampling-based trajectory generation solver is used that borrows from a similar control-theoretic approach. This chapter is based upon the following paper:

- Higgins, J., Mohammad, N. and Bezzo, N., " A model predictive path integral method for fast, proactive, and uncertainty-aware UAV planning in cluttered environments," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021*, pp. 830-837.

Chapter 5: Data-Driven Occlusion-Aware Navigation via Online Quantile Temporal Difference Learning

This chapter introduces a novel approach that casts the occlusion-aware navigation problem as a risk-aware navigation problem, borrowing from the concepts established in Chapter 4. This work explores alternative methods of viewing the occlusion-aware problem beyond simply avoiding collisions, and instead focuses on other benefits towards occlusion-aware navigation, such as time-optimality in the face of dynamic obstacle uncertainty within occluded regions. This is accomplished through the creation of a risk metric that encodes the probability of encountering a general negative outcome, which is defined in this work as sensing a dynamic obstacle emerge from an occluded region close to the robot, although it may potentially be applied to other use cases. This risk metric is constructed from experience gathered by the ego robot at run time, and *not* using pre-defined metrics, as was the case in the approaches of Chapter 2 and Chapter 3. This enables the approach to adjust to newly acquired data and create a policy bespoke to any environment in which it is deployed. Lastly, this approach leverages concepts borrowed from the reinforcement learning (RL) community and constructs this risk metric using Quantile Temporal Difference learning (QTD). This chapter is based on the following paper currently under review:

- Higgins, J. and Bezzo, N., "Data-Driven Occlusion-Aware Navigation via Online Quantile Temporal Difference Learning," *submitted and under review to IEEE Transactions on Robotics (T-RO).*

Chapter 6: What's the Worst That Can Happen? Run Time Data-driven Occlusion-Aware Navigation

In this chapter, we detail our current work on occlusion-aware navigation. As with the work in Chapter 5, the current research effort involves the creation of risk-based policy that incorporates and adjusts to data collected at run time. While Chapter 5 focuses on avoiding negative outcomes experienced by the ego robot, this current work instead uses collected data observed around occlusions to infer what *could* happen before the ego robot actually arrives at the occlusion. Additionally, this work provides a novel way of framing the risk-based path planning problem as a travel time minimization problem, in which the uncertainty of dynamic obstacles within occluded regions causes potential uncertainty in the travel-time required to reach a goal. A risk metric is defined over the distribution of possible travel times, and a graph-search algorithm is used to find a path that minimizes this risk metric. This chapter is based on the following current work:

- Higgins, J. and Bezzo, N., "Data-driven Occlusion-Aware Navigation at Run-time," *in preparation for submission to the 2025 International Conference on Robotics and Automation (ICRA).*

Chapter 7: Conclusions and Future Work

In this chapter, we conclude the dissertation by summarizing the results from all the aforementioned works and discuss 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 autonomous robotic occlusion-aware navigation by providing:

- A novel occlusion-aware control framework for common human-populated environments, quantifying the amount of FOV occluded by a hallway corner and actively seeking to minimize this occlusion while enforcing safety and liveliness.
- Extending this work to unstructured and unknown environments populated by generally-shaped occlusions, providing a generalized way to estimate the portion of the FOV that is occluded by an obstacle.
- Development of a novel data-informed risk metric that is efficiently learned from relatively small amounts of data and, when minimized, produces trajectories that reduce risk.
- Development of a technique to optimize over non-differentiable costs at run time, allowing to use optimization problems of a more general form that can tackle more complex scenarios.
- A novel approach that casts the occlusion-aware navigation as a risk-aware navigation problem, investigating other benefits to occlusion-aware navigation rather than purely a safety-based consideration.
- A novel framework for learning this risk-based occlusion-aware policy at run time from the data directly experience by the ego robot, leveraging techniques borrowed from RL to adjust the policy bespoke to a particular environment.
- Development of a novel occlusion-aware motion planning approach that leverages historical observations of dynamic obstacles to infer the risk associated with certain environmental occlusions.
- Formalizing a novel approach to cast the occlusion-aware navigation problem as a risk-based navigation problem, which associates the uncertainty created by occlusions with a risk of increased travel time, and actively finding a path that minimizes this risk.

Part I

Visibility-Based Occlusion Navigation

Chapter 2

Negotiating Visibility for Safe Autonomous Navigation in Occluding and Uncertain Environments

2.1 Introduction

From food delivery to warehouse logistics, many AMR are increasingly being developed for and deployed in structured environments such as office buildings or street sidewalks. These structured environments are usually characterized by floor plans with regularly spaced hallways, rectangular rooms and hallway intersections. This chapter presents an approach that leverages this structure to reason about the visibility from certain positions within the environment. The proposed control policy combines both visibility and safety constraints into a single framework, using these objectives to inform the importance of the other in an intuitive and natural way. This control framework is shown in Fig. 2.1 and captures the complex notions of safety and visibility, yet is kept simple so that it may be easily deployed by efficiently leveraging data easily obtainable from any standard line-of-sight sensor.

The main thrust of this work is to cast the promotion of visibility as an optimization problem, in which the effect of occlusions are actively minimized by the controller. This is achieved by quantifying the geometric area that is within the robot's FOV radius but is occluded by an obstacle. Fig. 2.2 shows a motivating situation in which the robot is tasked with moving towards a goal

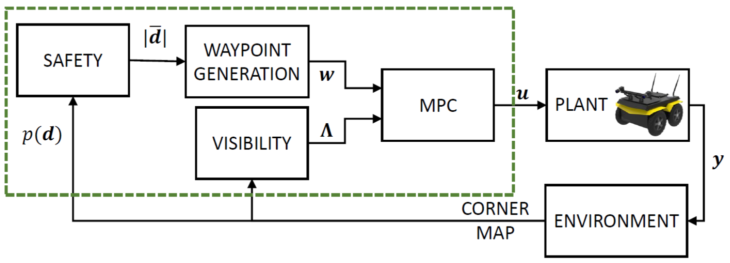


Figure 2.1: Block diagram of the visibility-based occlusion-aware motion planning approach.

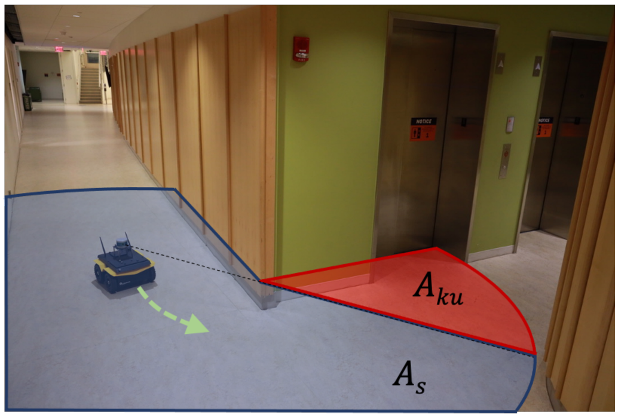


Figure 2.2: Pictorial representation of the problem covered in this work. A_s represents the visible region while A_{ku} is the "known-unknown" not visible sensed area by the robot.

location that lies behind an occluding corner. As the robot approaches this occluding corner, the line-of-sight sensors are able to detect an area A_s, while a portion of the environment known to be traversable is still left unobservable from the current position of the robot. We label this geometric occluded area as the "known-unknown" area A_{ku}, and seek to minimize this area as the robot navigates an environment, thus promoting higher perception around occlusions, rather than cutting around occluding corners. In addition to this perception objective, safety is also considered within the proposed framework as a means of ensuring the robot avoids collisions as it travels throughout the environment.

This work presents two main contributions: (i) the formulation of an analytic perception objective that, when used in the cost function of an optimal control problem, results in motion that increases perception around occlusions, and (ii) an occupancy grid-based technique to determine if the robot can safely navigate around the occlusion in the presence of traffic from other moving actors. This work was published in a journal (RAL'21), as well as presented as a conference paper (ICRA'21).

2.2 Problem Formulation

In this work we are interested in finding a control policy for a robot to negotiate occlusions while considering visibility, safety, and speed constraints. The framework should be able to apply to any general autonomous mobile system, as well as be able to generalize to any structured occluded environment. While there are many ways of approaching this multi-faceted problem, our approach decouples this research question into two sub-problems whose solution strongly affects the overall motion of the autonomous system. Formally, these two problems can be defined as follows:

Problem 1: ***Safe Navigation*:** A robot must be able to avoid collision with any obstacle within its visibility range at any time over a time horizon t_T, or mathematically:

$$||\boldsymbol{p}(t') - \boldsymbol{o}_i|| > 0, \forall t' \in [t, t + t_T], \forall i \in [1, n_o] \tag{2.1}$$

Here, $\boldsymbol{p}(t) = [x, y]^\intercal$ is the position of the robot, $\boldsymbol{o}_i(t) = [o_x, o_y]^\intercal$ is the position of the i^{th} obstacle in the $x - y$ plane at time t, and n_o is the number of obstacles. In the context of this work, this problem implies that the robot must be able to stop and avoid a collision with actors and obstacles that are occluded, and hence may suddenly appear in in the FOV of the vehicle.

This work also seeks to solve the problem of visibility-aware navigation. Motivated by Fig. 2.2, we define a known-unknown region $A_{ku}(t)$ as the area within a FOV radius $A(t)$ minus the area $A_s(t)$ not occluded by obstacles in the environment, or $A_{ku}(t) = A(t) - A_s(t)$. With this in mind, the second problem that we propose to solve in this work is:

Problem 3.2: ***Minimizing Known-unknown Occlusions*:** Given the safety constraint defined in Problem 3.1, find a control policy $\mathcal{P}_u$ which at runtime maximizes visibility in an occluded environment, or equivalently minimizes $A_{ku}(t)$:

$$\mathcal{P}_u(t) = \arg\min_{\boldsymbol{u}} A_{ku}(t), \forall t \tag{2.2}$$

where $\boldsymbol{u}$ is the commanded input to the robot.

2.3 Approach

2.3.1 MPC-based Known-Unknown Minimization

In this work, we are interested in designing a single model predictive controller (MPC) framework that negotiates occluded intersections of varying shapes and sizes. MPC operates on solving an online optimal control problem (OCP) that optimizes over a moving prediction horizon. This OCP requires two main components: (1) a cost function J that the OCP minimizes at each time step, and (2) feasibility regions that the OCP must respect. The cost function J at time t for our specific problem is expressed as:

$$\begin{aligned} J = &(\boldsymbol{x}_{t+T} - \boldsymbol{w}_t)^\intercal \boldsymbol{Q}_{t+T}(\boldsymbol{x}_{t+N} - \boldsymbol{w}_t) \\ &+ \sum_{i=1}^{T-1} (\boldsymbol{x}_{t+i} - \boldsymbol{w}_t)^\intercal \boldsymbol{Q}_{t+i}(\boldsymbol{x}_{t+i} - \boldsymbol{w}_t) + \boldsymbol{u}_{t+i-1}^\intercal \boldsymbol{R} \boldsymbol{u}_{t+i-1} \\ &+ w_\Lambda \Lambda^2(x_{t+i}, y_{t+i}) \end{aligned} \tag{2.3}$$

where $\boldsymbol{x}_i$ is the state space vector for the ith prediction step, $\boldsymbol{w}_t$ is the desired reference, and $\boldsymbol{u}_i$ is the ith control input that the MPC computes at each sampling period. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the cost weighting matrices for state space position and control input reference tracking, respectively.

The OCP is then formulated as:

$$\begin{aligned}
\arg\min_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{N-1}} \quad & J(\boldsymbol{x}_0, \boldsymbol{u}_0, \ldots \boldsymbol{u}_{N-1}) \\
\text{subj. to} \quad & \boldsymbol{u}_{t+i} \in \mathcal{U}_t, \quad \forall i = [0, N-1] \\
& \boldsymbol{x}_{t+i} \in \mathcal{X}_t, \quad \forall i = [1, N]
\end{aligned} \tag{2.4}$$

where $\boldsymbol{x}_{t+i}$ is the model-based predicted state at future time $t + i \times \Delta t$, Δt is the sampling time and T is the prediction horizon over N steps. The feasibility regions for the control inputs and state-space variables are denoted as $\mathcal{U}_t$ and $\mathcal{X}_t$, respectively. In this chapter, control inputs are restricted by simple min-max inequalities $\boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max}$. The feasibility region for the position component of the state is denoted by $\mathcal{P}_t$ and defined by an H-Polyhedron:

$$\mathcal{P}_t = \{\boldsymbol{p} \in \mathbb{R}^2 : A\boldsymbol{p}(t) \leq b\} \tag{2.5}$$

where the matrices A and b that define $\mathcal{P}_t$ depend on the position of the robot $\boldsymbol{p}(t)$, as well as the width of the current corridor w_0 and the width of the next corridor w_1:

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \\ 1 & -y/x \end{pmatrix}; \quad b = \begin{pmatrix} w_0 \\ w_1 \\ 0 \end{pmatrix} \tag{2.6}$$

Motion through the environment is produced by a pure pursuit approach of a waypoint $\boldsymbol{w}_t$ that changes over time. The novelty about our MPC framework is the inclusion of a new term $\Lambda(x, y)$ in (2.3), which defines the perception objective. The purpose of the perception objective $\Lambda(x, y)$ is to value the current position based on this occluded area.

An exact analytical expression for the known-unknown area $A_{ku}(t)$ of an occluded environment is in general a piecewise continuous function that is not differentiable and computationally intractable. Instead, this work finds use in a simple analytical expression that closely correlates to $A_{ku}(t)$. This analytical expression is defined by distances Δx and Δy relative to the occluding corner, shown in Fig. 2.3. The angle between the position of the robot and the upcoming corridor is defined as $\theta = \arctan(\Delta y/\Delta x)$. With these quantities, the perception objective is defined as:

$$\Lambda(\Delta x, \Delta y) = \frac{\theta}{\Delta y} = \frac{\arctan(\Delta y/\Delta x)}{\Delta y} \tag{2.7}$$

In other words, the perception objective is defined as the ratio between the angle θ and the

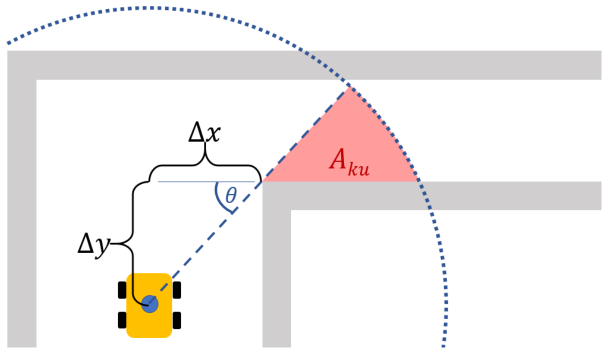


Figure 2.3: Defining the perception objective in terms of the occluding corner, relative to the current location of the robot.

parallel distance Δy. This expression has two main properties that make it appealing to use as a perception objective:

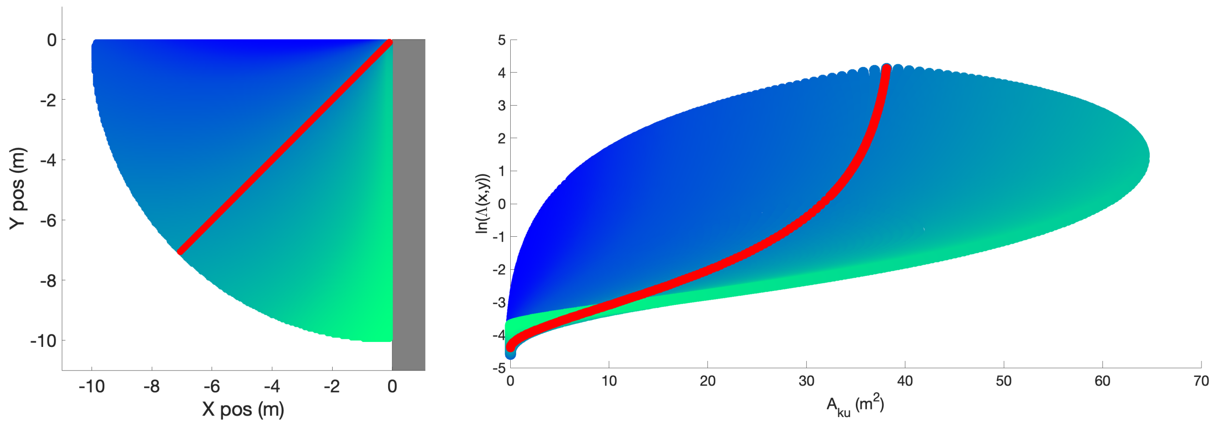
- $\lim\limits_{\Delta y \longrightarrow \infty} \Lambda(\Delta x, \Delta y) = 0$, meaning that this perception objective naturally tends to zero if the robot is far away from the occluding corner. In this case, the robot's motion will not be affected by the perception objective.

- $\lim\limits_{\Delta y \longrightarrow 0, \Delta x \neq 0} \Lambda(\Delta x, \Delta y) = 1/\Delta x$, meaning that as the robot approaches the occluding corner, the perception objective takes on the value $1/\Delta x$ and the MPC tries to increase Δx in order to minimize the perception objective. By increasing Δx, the occluded area is naturally minimized.

Lastly, it may be noted that the relative position of occluding corner $(\Delta x, \Delta y)$ may be easily identified using the range measures provided by any standard line-of-sight sensor. This is accomplished by noting any large jumps in range value between two successive range angles, and identifying the close range location as the occluding corner.

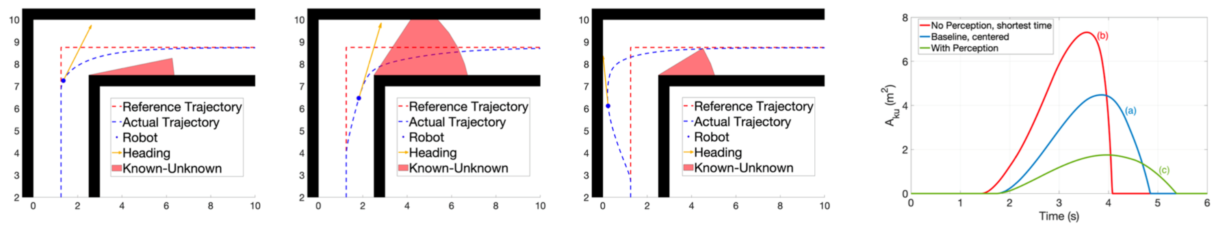
We will now show that the analytical expression in (2.7) mimics the desired behavior that an ideal perception objective would have and provides a differentiable cost function that any nonlinear MPC can handle. Fig. 2.4 shows a mapping between spatial points occluded by an upcoming corner, the desired value $A_{ku}(x, y)$ and perception objective $\Lambda(x, y)$. The values of $A_{ku}(x, y)$ are considered the "ground truth" that the analytical perception objective $\Lambda(x, y)$ is approximating. The quantity $\log(\Lambda(x, y))$ is plotted against $A_{ku}(x, y)$ to address the non-linear relationship between them. Highlighted in Fig. 2.4 is a red line of locations along which $\ln(\Lambda(x, y))$ and $A_{ku}(x, y)$ share a correlation of 0.96, implying a strong relationship between the two values. In other words, as the MPC decides to increase/decrease the perception objective as defined in (2.7), this correlates to an increase/decrease in the known-unknown area.

Analyzing the effects of the perception constraint in (2.3), we note that a large weight w_Λ causes the MPC to return commanded inputs whose overall effect is to provide a better vantage down



(a) (b)

Figure 2.4: Mapping points in the $x - y$ plane that are occluded by a corner (a) to values of the known-unknown area $A_{ku}(x, y)$ and the logarithm of the perception objective $\Lambda(x, y)$ (b).



(a) (b) (c) (d)

Figure 2.5: Example motion showing the effects of motion that reduces the known-unknown area.

occluded hallways. Fig. 2.5 shows the effect of this perception objective. First, Fig. 2.5(a) shows motion of a robot moving with a baseline policy of following the middle of the hallway. Fig. 2.5(b) instead shows a minimum-time policy that seeks to quickly cut around the corner, with $w_\Lambda = 0$. In contrast to these policies, Fig. 2.5(c) shows the resulting trajectory with $w_\Lambda > 0$ in which the robot moves to decrease A_{ku}, thereby increasing visibility. Finally, Fig. 2.5(d) shows the value of A_{ku} over time for each trajectory, quantitatively demonstrating the effects of each trajectory.

2.3.2 Safety Constraint

As mentioned in Sec. 2.3.1, the MPC is given a waypoint $\boldsymbol{w}(t)$ that serves as a pure pursuit objective for the robot to follow. In the proposed framework, safety is enforced by adapting $\boldsymbol{w}(t)$ depending on the uncertainty of the upcoming environment.

In the presence of uncertainty due to possible incoming traffic of actors from around occlusions, safety is determined through probabilistic means by computing the average distance until collision. In order to calculate this expectation value, one must know the probabilities of collision for different locations in space at particular instances in time. While this may seem unwieldy at first, such a task becomes more manageable by making the following connection: *the probability of colliding with an object at a given point in space is the same as that location's probability of occupancy.* Occupancy grids are a common and well-studied approach for mapping [83], and readily give the probability of occupancy for any grid location. The proposed framework borrows this idea of occupancy-for-mapping and instead applies it towards calculating the likelihood of collision in an occluded hallway.

In choosing a waypoint $\boldsymbol{w}$ in an uncertain environment, it is beneficial to have a function that directly connects to the safety of $\boldsymbol{w}$. In this framework, this function is the expected distance to collision beyond the corner and is calculated using estimated future probabilities of occupancy in the upcoming section of the hallway.

Define $\Delta\boldsymbol{d} = \boldsymbol{p} - \boldsymbol{w}$, and divide $\Delta\boldsymbol{d}$ up into n sufficiently small line segments $\boldsymbol{d}_i = \boldsymbol{p} + \frac{i}{n}\Delta\boldsymbol{d}$, $i = 1, \ldots, n$. Additionally, let $\mathcal{S}_{\text{next}} \in \mathbb{R}^2$ define the area of the upcoming hallway. Define the set $\mathcal{D}$ as,

$$\mathcal{D} = \{\boldsymbol{d}_i : \boldsymbol{d}_i \in \mathcal{S}_{\text{next}}\} \tag{2.8}$$

In other words, $\mathcal{D}$ contains all points $\boldsymbol{d}_i$ that lie beyond the occluding corner.

An occupancy grid map takes as input the $x - y$ position in space and returns the probability that a region is occupied by an obstacle. At each sampling time, this occupancy grid is updated by what is observed by the robot. Define $p_t(\boldsymbol{d}_i)$ as the current probability function, with $p_t(\boldsymbol{d}_i) = 1$ representing the knowledge that $\boldsymbol{d}_i$ is occupied by an obstacle with certainty. In order to address situations with dynamic obstacles (e.g., a person walking into view from around the corner), future occupancy $p_{t+\Delta t}(\boldsymbol{d}_i)$ at some time Δt later is predicted from $p_t(\boldsymbol{d}_i)$ via convolution with a probabilistic motion model. For ease of discussion, the probabilistic motion model is assumed to be a simple uni-directional motion down the corridor. The parameter Δt is estimated as the distance to $\boldsymbol{w}$ divided by the speed the robot is currently moving in that direction:

$$\Delta t = \frac{|\boldsymbol{d}|}{\dot{\boldsymbol{p}} \cdot \hat{\boldsymbol{d}}^{\intercal}} \tag{2.9}$$

The expected location of collision $\overline{\boldsymbol{d}}$ can be defined as:

$$\overline{\boldsymbol{d}} = \sum_{i \in \mathcal{D}} \boldsymbol{d}_i p_{t+\Delta t}(\boldsymbol{d}_i) \Pi_{j=1}^{i} \left[1 - p_{t+\Delta t}(\boldsymbol{d}_j)\right] \tag{2.10}$$

Here, (2.10) can be thought in terms of a generalized geometric probability distribution, in that the term $p(\boldsymbol{d}_i)\Pi_{j=1}^{i}\left[1 - p(\boldsymbol{d}_j)\right]$ is interpreted as the probability that the robot travels along $\Delta\boldsymbol{d}$ from its current position $\boldsymbol{p}$ and collides with an obstacle only at $\boldsymbol{d}_i$. It should be noted that (2.10) is a

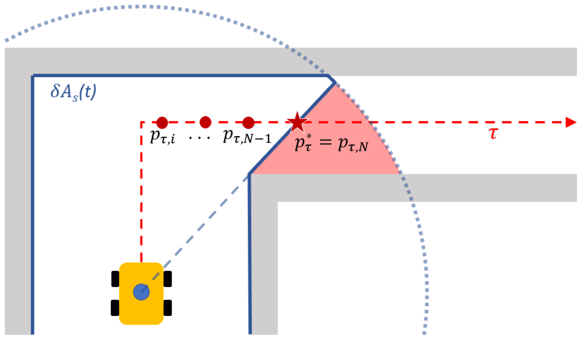


Figure 2.6: Waypoint placement in safe and unsafe situations.

conservative approximation of the actual expected distance to collision, as the motion of the robot may not be along $\boldsymbol{d}$.

The safety of a waypoint $\boldsymbol{w}$ is determined by comparing the expected distance to collision $|\bar{\boldsymbol{d}}|$ to the stopping distance of the robot. If we assume that the motion of the robot is limited to a maximum velocity $|\dot{\boldsymbol{p}}|_{\max}$ and acceleration $|\ddot{\boldsymbol{p}}|_{\max}$, a maximum stopping distance can be computed as:

$$d_{\text{stop, max}} = \frac{|\dot{\boldsymbol{p}}|^2_{\max}}{2|\ddot{\boldsymbol{p}}|_{\max}} \tag{2.11}$$

A waypoint $\boldsymbol{w}$ is considered safe when the expected distance to collision is greater than the maximum stopping distance. In this way, the robot has enough room to stop completely before its anticipated collision. To indicate safety we introduce a binary variable $\Theta = 1(0)$ when safe (unsafe):

$$\Theta = \begin{cases} 1, & \text{if } |\bar{\boldsymbol{d}}|/d_{\text{stop, max}} > 1 \\ 0, & \text{else} \end{cases} \tag{2.12}$$

Consider now a predefined desired trajectory $\boldsymbol{\tau}$ used as a means of routing the robot around known static obstacles, such as walls, towards the robot's goal position. Points along $\boldsymbol{\tau}$ that are currently visible to the robot are considered as candidate waypoints, with (2.12) used to ensure that any particular candidate is safe to move towards.

Let $A_s(t)$ define the area visible to the robot at time t, and define the boundary of this area as $\delta A_s(t) \in \mathbb{R}^2$ as depicted in Fig. 2.6.

Let $\boldsymbol{\tau}' = \{\boldsymbol{p}_{\tau,0}, \ldots, \boldsymbol{p}_{\tau,i} \ldots, \boldsymbol{p}_{\tau,N}\}$ with $i = \{1, \ldots, N\}$ be an indexed set of N discrete points that make up the visible model trajectory $\boldsymbol{\tau}'$. The waypoint location $\boldsymbol{w}$ is placed at the furthest point along $\boldsymbol{\tau}'$ that is considered safe according to (2.12). The first point considered is the farthest visible point $\boldsymbol{p}_{\tau,N} = \delta A_s \cap \boldsymbol{\tau}$. If $\boldsymbol{p}_{\tau,N}$ is unsafe, then a closer visible point is considered, $\boldsymbol{p}_{\tau,N-1}$. Fig. 2.6 shows how this process may be repeated until a safe waypoint $\boldsymbol{w} = \boldsymbol{p}_{\tau,N-k}$ is found. If no point on $\boldsymbol{\tau}$ is considered safe in the upcoming corridor, the waypoint is placed at the entrance of the upcoming corridor $\boldsymbol{p}_{\tau,0}$ and the upcoming corridor is considered unsafe. In situations where $\delta A_s \cap \boldsymbol{\tau} = \emptyset$, the waypoint is chosen as the closest point $\boldsymbol{v}^* \in \delta A_s$ to the desired trajectory $\boldsymbol{\tau}$.

$$\boldsymbol{w} = \begin{cases} \boldsymbol{p}_{\tau,N-k}, \quad k \in [1, N] & \text{if } \delta A_s \cap \boldsymbol{\tau} \neq \emptyset \\ \boldsymbol{v}^* \in \delta A_s \ s.t. \ |\boldsymbol{p} - \boldsymbol{v}^*| \leq |\boldsymbol{p} - \boldsymbol{v}| & \\ \qquad\qquad \forall \boldsymbol{v} \in \delta A_s(t), & \text{else} \end{cases} \tag{2.13}$$

When the waypoint is placed at $\boldsymbol{w} = \boldsymbol{p}_{\tau,0}$, it means that there is some non-negligible probability of expected traffic coming from around the occluding corner. In this situation, it is advantageous for the robot to gain visibility around the occluding corner to increase the certainty that the upcoming hallway is unoccupied, thereby lengthening the expected distance to collision in (2.12). Thus the perception objective is activated and the waypoint is fixed at $\boldsymbol{p}_{\tau,0}$. By bringing the waypoint closer to the position of the robot, a desired side effect is that the MPC will reduce the speed of the robot to a safe stop at the selected waypoint if needed.

2.4 Simulations

The first case study investigated in this work is a simple "L"-shaped corridor with one occluding corner. The simulated UGV robot uses a common differential drive motion model [48] and has a maximum velocity of 2 m/s and maximum acceleration of 1 m/s^2. Thus, from (2.11), $d_{\text{stop, max}} = 2$ m. The FOV of the UGV is limited to a 5 m radius. The optimal control problem was solved using ACADO toolkit [39] and qpOASES [32] as the solver. Fig. 2.7 shows a series of snapshots of the proposed framework navigating the robot through the occluded intersection. The gray grid cells represent the estimated future probabilities of occupancy. Current occupancy probabilities are

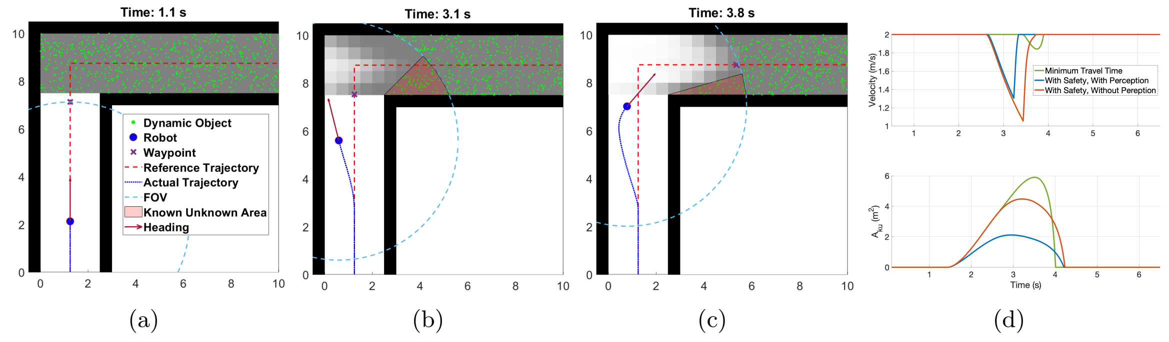


Figure 2.7: Snapshots of a simulation case study in which a UGV navigates an occluding corner, considering uncertainties. The gray shaded region on the second corridor indicates the probability that a certain cell is occupied by other actors. (d) shows plots of velocity and A_{ku} for our policy framework with and without perception compared to a minimum travel time implementation.

determined through a log-odds Bayesian update, and used to estimate future occupancy probabilities by convolving with a known motion model.

Also shown in Fig. 2.7 are simulated dynamic objects, uniformly ranging in speed from 0 to 5 m/s traveling to the left. The convolution model used to temporally propagate the occupancy probabilities correlates to this uniform probability in dynamic obstacle velocity. As the simulation ran, the UGV could "sense" a dynamic object only when it was within the UGV's FOV, recording its distance. This distance serves as a conservative estimate on safety of the proposed framework since it is designed to provide ample reaction time in uncertain situations, and not provide a policy that plans around dynamic obstacles. For these reasons, the dynamic obstacles are removed from simulation when they are first observed.

Fig. 2.7(d) plots velocities and known-unknown areas for these simulations for comparison. It is apparent from Fig. 2.7(d) that the proposed policy framework performs best at maximizing visibility. What may be surprising is that by considering visibility constraints, the proposed framework *also moves faster around the corner than motion guided only by the safety module*. The UGV is able to do this because having visibility around the occluding corner helps reduce the uncertainty in occupancy, establishing safety more quickly than using the safety module alone.

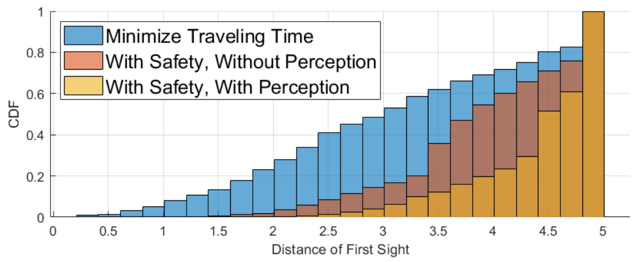


Figure 2.8: The cumulative distribution function of the distance that the robot first senses a dynamic object while negotiating the L-shaped corridor.

Fig. 2.8 shows these results as a cumulative distribution function over the distance that a dynamic obstacle was first sensed. As a point of comparison, also shown are the results of a UGV moving to minimize traveling time (i.e., quickly cutting the corner) as well as a UGV following the safety module without a perception objective to help with visibility. The figure shows how motion that minimizes traveling time has a 23% chance of sensing a dynamic obstacle under $d_{\text{stop, max}} = 2$ m., which is unacceptable in a safety-critical situation. With the full safety and perception framework, there was no situation where the UGV sensed a dynamic obstacle within its maximum stopping distance.

The second case-study focused on a real-world situation in which an industrial UGV was tasked to navigate a series of hallways inside a warehouse to retrieve an item from a stockroom and take it to a specified location. In this case study, the UGV must reach the stockroom via a main hallway that is often occupied by dynamic obstacles (e.g., people, other robots), and thus has some uncertainty of occupancy. Two scenarios were tested: (1) the main hallway is known to be clear of dynamic obstacles (e.g. it is night-time and no other actors are present in the warehouse) and thus this main hallway is known to be safe a priori, and (2) occupancy in the main hallway is unknown, with a probabilistic motion model that assumes all dynamic obstacles move down the hallway. Fig. 2.9 shows the setup and results for these scenarios.

As Fig. 2.9 shows, the main difference of trajectories between the two scenarios is when the UGV enters the main hallway. When there is uncertainty, the UGV moves to gain visibility up the hallway, and when the main hallway is known to be safe, it instead cuts the corner. Fig. 2.9(e) shows how the known-unknown area is reduced when occupancy is uncertain in the main hallway.

2.5 Experiments

Experimental validations were performed with a Clearpath Robotics Jackal UGV inside our lab. As a proof of concept, different occluded geometries were created to showcase how the proposed framework adapted to different scenarios. For each scenario, a Vicon motion capture system was used to measure state-space values of the Jackal, which were fed into the policy framework outlined above. The MPC executed at 10 Hz, producing commanded velocities which were fed to a lower level controller executing at 100 Hz. As the Jackal can follow commanded velocities, it can instantly stop moving at any point and $d_{\text{stop, max}}$ is effectively zero. Because of this feature, only the effect of the perception constraint on motion was explored in experiments.

Fig. 2.10 shows snapshots for the case study of a UGV approaching the intersection of two hallways where its sensing capabilities are occluded by a sharp corner. Two different objectives were tested: (1) minimum time and (2) perception. Fig. 2.10 shows snapshots of these two experiments, as well as known-unknown areas recorded by the Jackal. Additionally Figs. 2.10(a,b) show laser

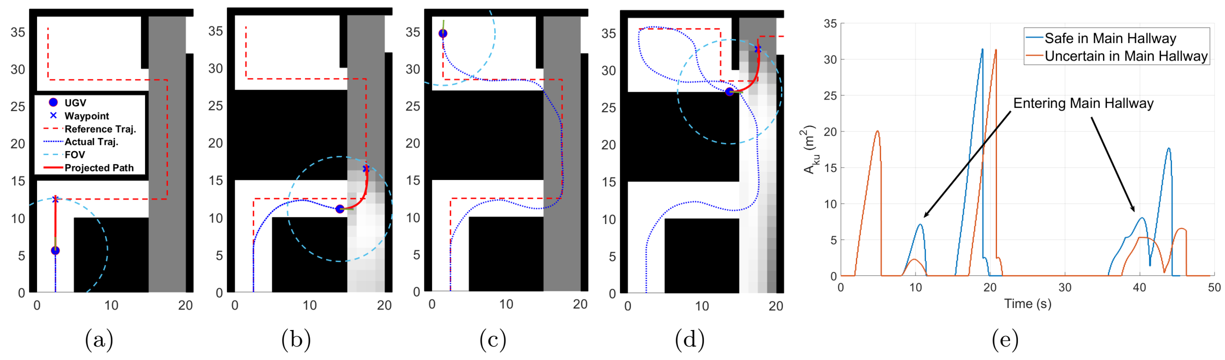
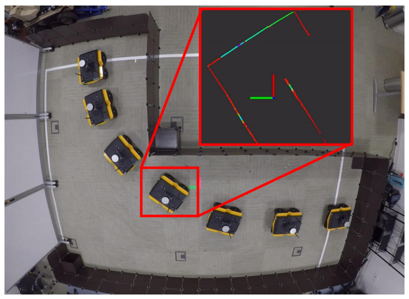
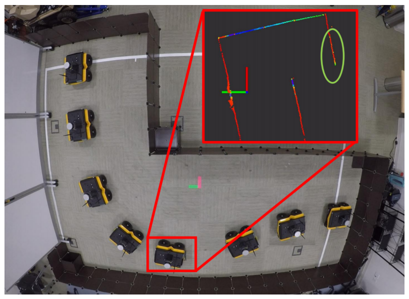
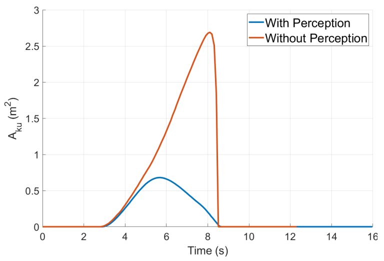


Figure 2.9: Sequence of snapshots for a simulation of a robot operating in a occluding environment with variable expected traffic of dynamic objects (a-d). In (e) it is depicted the comparison between the known-unknown areas over time of the case where the long hallway is safe vs unsafe.


(a) Motion with minimum traveling time.


(b) Motion with perception.



(c) A_{ku} over time

Figure 2.10: Snapshots of experiments and data for the two-corridor scenario. Highlighted in (a) and (b) are the lidar point-cloud data before passing the corner showing a decreased known-unknown in (b). In (c) the plots show the difference in A_{ku} between (a) and (b).

scan data recorded by the onboard lidar. The impact of including perception is highlighted by the additional laser scan points around the occluding corner.

2.6 Discussion and Conclusion

This chapter presents an approach for visibility-based occlusion-aware navigation in structured environments. This was achieved by using the relative position of the occluding corner as a means of approximating the occluded "known-unknown" area within a structure environment. This analytical approximation was placed within the cost function of an OCP, acting as a perception objective that, when minimized, encouraged motion that increased visibility around these occluding corners. When paired with a safety module, the combined framework allowed for motion that maintained speed while negotiating an occluded environment. The proposed approach was validated through both simulations and experiments.

Due to the simple analytical form provided by the perception objective, occluding corners within structure environments are relatively easy to locate using any standard line-of-sight sensor, meaning this approach can leverage the data generated by these sensors to inform the occlusion-aware motion policy without the need for any training. Thus, the approach presented in this chapter presents a data-efficient method for occlusion-aware navigation.

The main limitation of this approach is the need for tuning of cost function weights within (2.3). These weights are parameters whose values must be determined by an operator prior to deployment in the real world. For a single occluding corner, (2.3) includes nine parameters (assuming matricies $\boldsymbol{Q}_{t+T}$, $\boldsymbol{Q}_{t+i}$ and $\boldsymbol{R}$ are diagonal) and becomes a more complex tasks as additional occluding corners are included within the cost function. Additionally, there is no principled way to approach this tuning beyond trial and error. Thus, there is a trade-off in which the training required of other approaches is replaced by some operator tuning with the approach proposed in this chapter. Lastly, this approach relies on the structured nature of the occluded environment in order for the perception objective to accurately approximate the known-unknown area. Environments with amorphous or unstructured obstacles may present a challenge for the approach discussed in this chapter. Chapter 3 addresses these concerns by providing an additional perception objective that may be applied to both unknown and unstructured environments.

Chapter 3

A Model Predictive-based Motion Planning Method for Safe and Agile Traversal of Unknown and Occluding Environments

3.1 Introduction

One natural extension of the work presented in Chapter 2 is to consider general environments in which the occlusions are not necessarily defined by hallways in a structured environment. In order to accommodate this more complex task, the visibility objective $\Lambda(\cdot)$ must be reformulated so that it may be applied to general occluding surfaces, and not just structured hallway-like walls. Additionally, unknown environments will not have a pre-defined trajectory τ that may be used to defined a safe waypoint $\boldsymbol{w}$ for the robot to travel towards. Thus, this section provides two novel approaches to both visibility and safety that may to utilized in general and unstructured environments. Fig. 3.1 shows a diagram on this approach. Although similar to Fig. 2.1, the safety module and the perception module are improved with more general mechanisms to account for both safety and visibility. Additionally, the visibility objective presented in this chapter utilizes geometric features within the environment that may be easily extracted using any standard two-dimensional line-of-sight sensor. Thus, the approach discussed here leverages data generated by the light-of-sight sensors as a means of defining the visibility objective to be optimized over.

3.2 Problem Formulation

In this work, we are interested in finding a control policy that can swiftly navigate a robot towards a goal while avoiding collision with both known and unknown obstacles in the environment.

Problem 1: ***Safe Navigation in Occluding Environments*****:** A robot must be able to avoid collision with any obstacle within its visibility range at any time over a time horizon T, or mathematically:

$$|\boldsymbol{p}(t') - \boldsymbol{o}_i|^2 > r_i, \forall t' \in [t, t+T], \forall i \in [1, n_o] \tag{3.1}$$

in which $\boldsymbol{p}(t) = [x, y]^\intercal$ is the position of the robot, $\boldsymbol{o}_i(t) = [o_x, o_y]^\intercal$ and r_i are the position and radius of the i^{th} obstacle in the $x - y$ plane at time t, and n_o is the number of obstacles. In the context of this work, this problem implies that the robot must be able to stop and avoid a collision with actors and obstacles that are occluded, and hence may suddenly appear in the FOV of the vehicle.

This work defines agility as the ability of the robot to track a reference speed. Let v_r and $v(t) = |[\dot{x}(t), \dot{y}(t)]|^2$ represent the reference speed and speed of the robot at time t, respectively. The velocity error may then be defined as:

$$e_v(t) = |v(t) - v_r|$$

At the same time, motion should be commanded that promotes visibility around occlusions. Define $A(t)$ as the area within the FOV that is not known a priori to be occupied or blocked by obstacles, and $A_s(t) \in A(t)$ as the area that is visible to the robot. The difference between these two sets is the area that is occluded to the robot, which we quantify as the known-unknown area $A_{ku}(t) = A(t) - A_s(t)$. Fig. 1.1 shows an example of both A_s and A_{ku}.

Problem 2: *Promoting Speed and Visibility*: Given the safety constraint of Problem 3.1, find a control policy $\mathcal{P}_u$ that simultaneously minimizes both the velocity error and the known-unknown area at runtime:

$$\mathcal{P}_u(t) = \arg\min_{\boldsymbol{u}} \left(c_v e_v(t) + c_{ku} A_{ku}(t)\right), \forall t \tag{3.2}$$

where c_v and c_{ku} are hyperparameter constants of optimization and $\boldsymbol{u}$ is the commanded input to the robot.

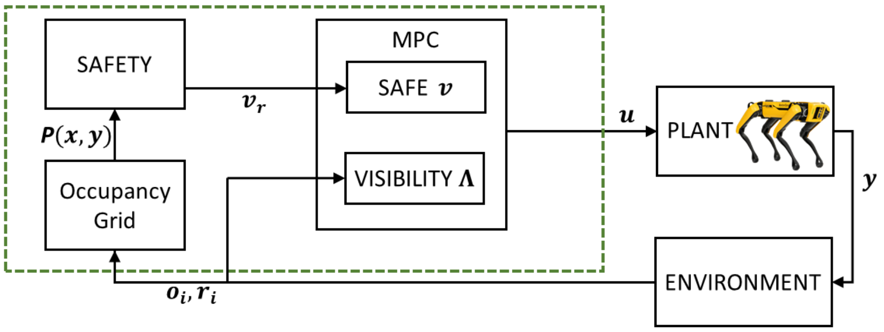


Figure 3.1: Diagram showing the different components of the proposed controller. The contributions of this section are within the green box.

3.3 Approach

3.3.1 MPC-based Motion Planning

In this work, an MPC commands motion of a system throughout the environment by solving an OCP online. Let us assume that the system follows general state dynamics defined by state $\boldsymbol{x}$ and control input $\boldsymbol{u}$:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{3.3}$$

The function $f(\cdot)$ is assumed to be time-invariant, meaning that the starting time may be taken as $t_0 = 0$ without loss of generality. Define $\boldsymbol{x}_r$ and $\boldsymbol{u}_r$ as the desired reference state and control input of the system, respectively. The cost function $J(\cdot)$ of this OCP can be defined over a future horizon T as:

$$\begin{aligned} J(\boldsymbol{x}(t), \boldsymbol{u}(t)) &= |\boldsymbol{x}(T) - \boldsymbol{x}_r|^2_{\boldsymbol{Q}_N} + \\ \int_0^T & |\boldsymbol{x}(t) - \boldsymbol{x}_r|^2_{\boldsymbol{Q}} + |\boldsymbol{u}(t) - \boldsymbol{u}_r|^2_{\boldsymbol{R}} + \sum_{i=1}^{n_{\text{obs}}} |\Lambda_i(\boldsymbol{p}(t))|^2_{\boldsymbol{M}} dt \end{aligned} \tag{3.4}$$

Here, $\boldsymbol{Q}_N$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite weighting matrices and $\boldsymbol{M}$ is a positive semi-definite scalar. These weights determine the relative importance in minimizing each term in the cost function. In addition to reference tracking, this cost function also includes a perception objective $\Lambda_i(\cdot)$ that is dependent only on the $x - y$ position of the robot, defined as $\boldsymbol{p}(t) = [x(t), y(t)]$, and the position of the nearest n_{obs} obstacles. By including this term, the resultant motion also seeks to minimize the known-unknown area behind occlusions, thereby increasing visibility of the environment.

Constraints can be included inside an OCP so that the solution respects the physical limits of the controlled system, e.g., a limit on the speed of the system, or applied control input. These constraints are generally cast as inequalities in the following form:

$$g_{\text{lim}}(\boldsymbol{x}, \boldsymbol{u}) \leq 0$$

Constraints on the position of the system $\boldsymbol{p}$ may also be used to command motion that avoids obstacles in the environment. In this approach, two types of constraints are considered. The first are half-plane constraints, defined with matrix $\boldsymbol{A}$ and offset term $\boldsymbol{b}$:

$$g_{\text{hp}}(\boldsymbol{p}) = \boldsymbol{A}\boldsymbol{p} - \boldsymbol{b} \leq 0 \tag{3.5}$$

Half-plane constrains are used for obstacles shapes that have a large eccentricity, such as the wall of a long hallway. Obstacles with low eccentricity are instead approximated with circles that envelope their boundary. These circles are defined by their position $\boldsymbol{c}(t) = [c_x(t), c_y(t)]$ and radius r, and this constraint may again be cast as an inequality:

$$g_{\text{circ}}(\boldsymbol{p}) = r_i^2 - |\boldsymbol{p} - \boldsymbol{c}_i|^2 \leq 0, \quad \forall i \in \{1, ..., n_{obs}\} \tag{3.6}$$

To simplify notation, these inequality constraints may be combined into one function $g(\boldsymbol{x}, \boldsymbol{u}) = [g_{\text{lim}}, g_{\text{hp}}, g_{\text{circ}}]$. The OCP that the MPC solves may thus be defined in the following way:

$$\begin{aligned} \arg\min_{\boldsymbol{u}(t)} \quad & J(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \text{subj. to} \quad & \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ & \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\ & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \leq 0 \end{aligned} \tag{3.7}$$

3.3.2 Visibility Objective for Unstructured Environments

The purpose of the perception objective is to directly correlate with the value of the known-unknown area behind occluding obstacles. In this way, as the MPC minimizes the cost defined in (3.4), it simultaneously seeks to reduce the position tracking error (i.e, move towards a goal) while also reducing this known-unknown area. Ideally, the perception objective $\Lambda(\cdot)$ would provide an exact value for the known-unknown area. Unfortunately, the known-unknown area is often non-smooth as a function of the robot position $\boldsymbol{p}$, meaning such a function would be discontinuous in the first derivative and could not be used in many general solvers available today that rely on the continuity of the cost derivative to find a minimum.

Our approach instead defines $\Lambda(\cdot)$ as an approximation of the known-unknown area. Assume that an occluding obstacle at position $\boldsymbol{c}$ is characterized by a radius r. For general obstacles with low eccentricity, r could be the size of a circle that completely envelopes the obstacle. Assuming that the occlusion is within the field of view of the robot with radius r_{fov}, and that $d = |\boldsymbol{p} - \boldsymbol{c}|^2$ is the Euclidean distance between the robot and the occlusion, an approximation of the known-unknown area $\widehat{A_{ku}}$ can be defined as:

$$\widehat{A_{ku}} = \frac{r}{d}(r_{\text{fov}}^2 - d^2) \tag{3.8}$$

This approximation for A_{ku} is motivated in Fig. 3.2. First, A_{ku} is approximated as the difference between two circle sectors with the same angle θ. The larger circle has radius r_{fov}, while the small circle has radius d, so $A_{ku} \approx \frac{\theta}{2}(r_{fov}^2 - d^2)$. A second approximation is made by defining the sector angle θ to be the arc length of the smaller sector ($\approx 2r$) divided by the radius d, so that $\theta \approx 2r/d$.

While this is a reasonable approximation for A_{ku} of a circle, one issue is that $\widehat{A_{ku}} < 0$ if the obstacle is outside of the FOV radius, or $d > r_{\text{fov}}$. In fact, $\widehat{A_{ku}}$ is negatively unbounded as $d \to \infty$.

This of course does not reflect reality, as $A_{ku} = 0$ for all $d > r_{\text{fov}}$. In order to correct this, $\widehat{A_{ku}}$ is put into a ReLU function defined as $h_{\text{ReLU}}(x) = \max(0, x)$ [76].

In other words, $h_{\text{ReLU}}(x)$ returns the input when it is positive, and 0 whenever the input is negative. Unfortunately, h_{ReLU} is an inconsistent function, and so will not work with many available nonlinear program solvers [44]. For this reason, the proposed approach uses the softplus function $h_{\text{sp}}(x) = \ln(1 + \exp x)$ as a continuous approximation to the ReLU function. The full perception objective is thus defined to be:

$$\Lambda(\boldsymbol{p}) = \ln\left(1 + \exp\left(\widehat{A_{ku}}\right)\right) \tag{3.9}$$

Fig. 3.3 shows how (3.9) correlates to the actual known-unknown area behind an obstacle. Varying positions of the robot are colored next to an occluding obstacle, with the respective known-unknown area shown with the corresponding color. For each position, the actual value of the known-unknown area is recorded, along with the value of $\Lambda(\cdot)$. These value pairs are also plotted in Fig. 3.3. The result is a near-perfect direct correlation between the two quantities, with a correlation of 0.9996. This means minimizing $\Lambda(\cdot)$ directly correlates to minimizing the actual known-unknown area of a circular obstacle.

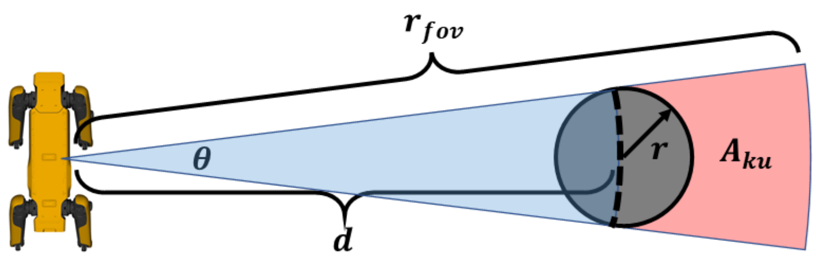
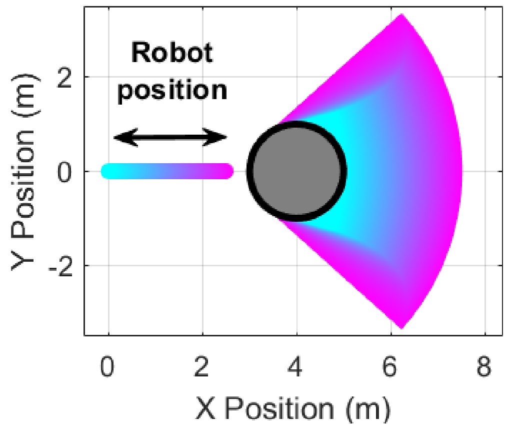
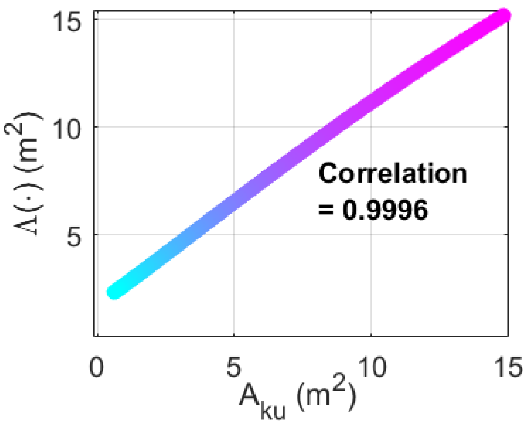


Figure 3.2: Known-unknown area A_{ku} for a circular obstacle within the FOV of the robot.



(a)



(b)

Figure 3.3: Correlation between perception objective $\Lambda(\cdot)$ and true known-unknown area A_{ku}.

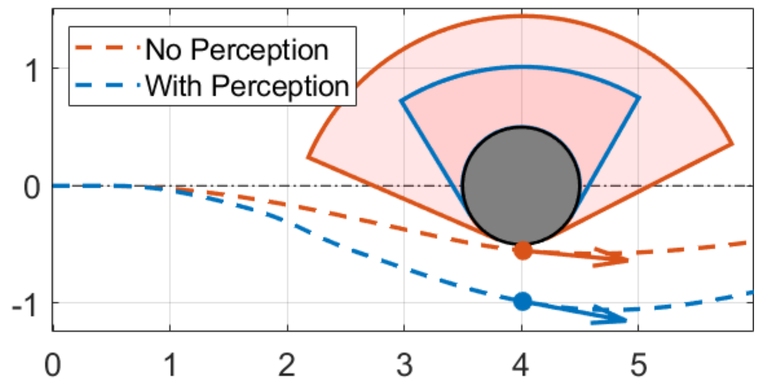


Figure 3.4: Comparison of motion commanded by the MPC, both with and without perception objective.

Finally, this perception objective can be used to approximate the known-unknown areas of multiple occlusions at once. The value of the perception objective for the i^{th} obstacle can be calculated for n_{obs} nearest obstacles. The square sum for each $\Lambda_i(\cdot)$ is then used for the total perception objective in (3.4).

Fig. 3.4 shows the effect that the perception objective has on the resulting trajectory of a robot whose goal is to the right of the map along $y = 0$ with one obstacle in the environment. Shown in orange is the trajectory of the robot without a perception objective; effectively, this means that $\boldsymbol{M} = 0$ in (3.4) and the MPC does not command motion to minimize the known-unknown area. In this case, the robot still avoids the obstacle, but comes very close to the obstacle boundary as it tries to minimize the distance to the $y = 0$ line. Shown in blue is the trajectory with the perception objective. Not only does the MPC command the robot towards the goal, but it also works to minimize the known-unknown area created by the occluding obstacle.

Although the approximate known-unknown area (3.8) assumes a circular occlusion, it may also be applied to corner occlusions with minimal adaptation. Consider the scenario shown in Fig. 3.5 where the robot approaches an occluding corner. The shape of the corner is assumed to be known a priori, e.g., the corner of a hallway in an office building where a map is available or identifiable using cameras and range sensors. The true known-unknown area is shown, as well as three virtual circles that are placed along the edge of the obstacle that is occluded. These virtual circles do not represent physical obstacles in the environment; instead, they are used to approximate the known-unknown area created by the corner occlusion.

In this case, the sum of perception objective for these three virtual circles $\sum_{i=1}^{3} |\Lambda(\cdot)|^2_{\boldsymbol{M}}$ approximates the true known-unknown area of the corner.

While the approximation becomes better with more virtual circles of smaller radii, this also increases the number of terms to be minimized in the objective function (3.4). This could affect solving time and convergence properties, so the appropriate number and size of virtual circles is

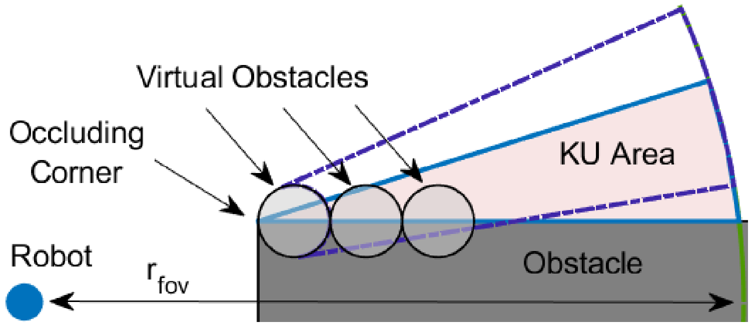


Figure 3.5: Pictorial representation of the approximation for the known-unknown area around a corner using virtual circles.

application dependent.

3.3.3 Safety Constraint for Unknown Environments

The constraints of the OCP defined in (3.7) can only guarantee collision-free motion for obstacles that are known to the robot. Thus, care must be taken to avoid obstacles that are occluded to the field of view of the robot. These unknown obstacles are located in the uncertain regions that are occluded to the robot or outside of its field of view

To mitigate this risk, a safety module is included that considers the distance the robot may travel until a collision is possible. With this distance, it determines a safe speed v_r for the robot to maintain. This speed is fed into the OCP of (3.7) as a reference speed for the MPC to track. The smaller the distance to collision, the slower v_r is set. This v_r is constantly recomputed at the same sampling rate as the MPC, allowing potentially rapid changes in the environment to be accurately accounted for in the MPC computation.

In order to estimate the distance to a possible collision, let us define the set of admissible controls as $\mathcal{U}$, so that $u(t) \in \mathcal{U} \, \forall t$. The forward reachable set (FRS) of a system is the set of state-space points that are reachable by the robot from its current state $\boldsymbol{x}_0$ within some time t. Assuming the state dynamics (3.3) are time invariant, the FRS may be defined as:

$$R^{n_x}(t, \boldsymbol{x}_0) = \left\{ \boldsymbol{x} \mid \boldsymbol{x}(t) = \boldsymbol{x}_0 + \int^t f(\boldsymbol{x}, \boldsymbol{u}) dt, \, \forall \, \boldsymbol{u} \in \mathcal{U} \right\} \tag{3.10}$$

Here, n_x is the dimension of the state space. In order to estimate collisions with other obstacles, let us define $R^2(t, \boldsymbol{x}_0)$ as the projection of $R^{n_x}(\cdot)$ onto the $x - y$ plane. For the remainder of the paper, $R^2(\cdot)$ will be referred to simply as the FRS, although (3.10) is the strict definition.

The FRS defines the portion of the environment that is accessible to the robot. In order to estimate a distance to collision within this FRS, an occupancy grid $P(\cdot)$ is defined for the entire environment [83]. This function $P(\cdot)$ takes in a location on the $x-y$ plane and returns the probability of occupancy at that location. Typically, $P(x, y) = 1.0$ denotes that an obstacle is known to be occupying space at location (x, y), $P(x, y) = 0.0$ denotes free space and $P(x, y) = 0.5$ denotes complete uncertainty, i.e., there is a 50% chance of an obstacle occupying that location.

This framework assumes that all known obstacles may be successfully avoided by appropriately defining the inequality constraints (3.5) and (3.6). Thus, the safety module is only concerned with the occluded space behind obstacles that *may* contain an obstacle. Mathematically, this uncertainty can be measured using the entropy of a binary variable $H(P) = -P \ln(P) - (1 - P)\ln(1 - P)$. Entropy of a grid square is maximum when $P(\cdot) = 0.5$, and decreases to zero as $P(\cdot) = 0.0$ or $P(\cdot) = 1.0$.

The distance to an unknown obstacle is defined as the distance between the robot and the nearest grid square within the FRS that has an entropy greater than some threshold $\underline{H}$:

$$\begin{gathered} d = |\boldsymbol{p} - \boldsymbol{q}^*| \text{ s.t. } |\boldsymbol{p} - \boldsymbol{q}^*| < |\boldsymbol{p} - \boldsymbol{q}| \,\forall\, \boldsymbol{q} \in R^2(t, \boldsymbol{x}), \\ H(\boldsymbol{q}) > \underline{H} \end{gathered} \tag{3.11}$$

Fig. 3.6(a) shows an example of this process. The FRS is shown in purple for a particular choice in robot dynamics (specifically for the figure, differential drive dynamics [83]). The occupancy map permeates the entire environment; white grid cells are known to be free of obstacles, while gray grid cells are uncertain because they have not been observed by the robot. The value for d is then chosen as the distance between the robot and the closest grid cell that i) is within the FRS and ii) has an entropy above a certain threshold.

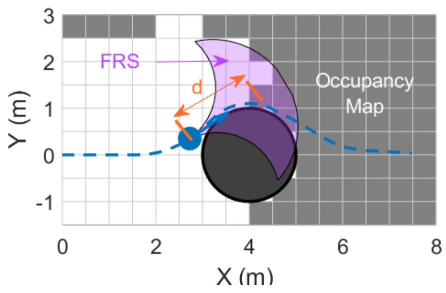
In order to consider dynamic obstacles, the future values of the occupancy map can be estimated by convolving the current occupancy map $P(\cdot)$ with a probabilistic motion model of the dynamic obstacles. This probabilistic motion model $P(\boldsymbol{o}'|\boldsymbol{o})$ defines the probability of a dynamic obstacle moving from location $\boldsymbol{o}$ to $\boldsymbol{o}'$ over some time dt. Convolution is applied over time dt to find a future occupancy estimate $P'(\cdot)$:

$$P'(\boldsymbol{o}') = \sum_{\boldsymbol{o}} P(\boldsymbol{o}'|\boldsymbol{o}) P(\boldsymbol{o}) \tag{3.12}$$

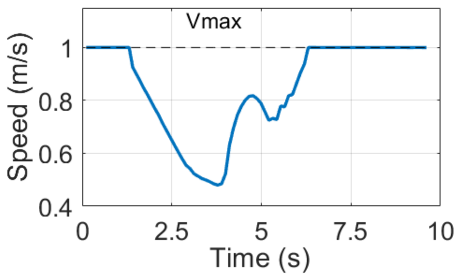
This operation is repeatedly applied until $P'(\cdot)$ estimates the occupancy grid at the same time t into the future as the FRS, effectively dilating the boundary of grid cells with high entropy and in turn decreasing d when the robot is near these boundaries.

This distance d is a conservative estimate on the required stopping distance of the robot. Assuming the robot has a maximum allowed acceleration $a_{\max}$, then the maximum allowed velocity can be found such that the robot moving with speed $v_{\text{stop}} = \sqrt{2da_{\max}}$ can stop within distance d under a constant acceleration $a_{\max}$.

There is only a need to slow down if v_{stop} is less than the maximum velocity of the system $v_{\max}$;



(a)



(b)

Figure 3.6: Example situation in which the safety module is used to command a slower velocity around an obstacle.

otherwise, it is beneficial to travel at or near $v_{\max}$ to reduce travel time. Thus, the following is sent as a velocity reference into (3.4), used in (3.7) as a reference speed for the MPC to track:

$$v_r = \min\left(v_{\text{stop}}, v_{\max}\right) \tag{3.13}$$

Fig. 3.6(b) shows the speed profile for the example in Fig. 3.6(a). As the robot moves close to the obstacle boundary, there is uncertainty of what lies behind the occlusion and the distance d decreases, in turn decreasing v_{stop}. The reference velocity v_r is determined through (3.13) and sent to the MPC, commanding a safe speed for the robot.

3.4 Simulations

Simulations were performed to show the capabilities of the proposed control policy to reduce the known-unknown area, increase speed and avoid collisions in unknown environments. In each simulation, a simple non-holonomic velocity motion model was assumed of the robot as $\{\dot{x} = v\cos\theta;\ \dot{y} = v\sin\theta;\ \dot{\theta} = \omega\}$, with pose $\boldsymbol{x} = [x, y, \theta]^\intercal$ controlled via commanded translational and rotational velocity $\boldsymbol{u} = [v, \omega]^\intercal$. To facilitate motion in general environments and prevent deadlock, an intermediary waypoint obtained by intersecting the robot's FOV with its desired path was given to the MPC for position tracking.

In order to solve the OCP posed in (3.7), the acados nonlinear program solver [85] was used for sequential quadratic programming, with qpOASES [31] as the underlying quadratic program solver. The prediction horizon $T = 5$ s, discretized into time steps $dt = 0.25$ s.

Fig. 3.7 shows an example with a robot moving clockwise around a square obstacle. The shape and location of this obstacle is known to the robot beforehand, e.g., it represents a static wall of an office building. Also present is a dynamic obstacle that is initially unknown and occluded to the robot. Fig. 3.7(a) shows motion without our approach in which timing and positioning cause the

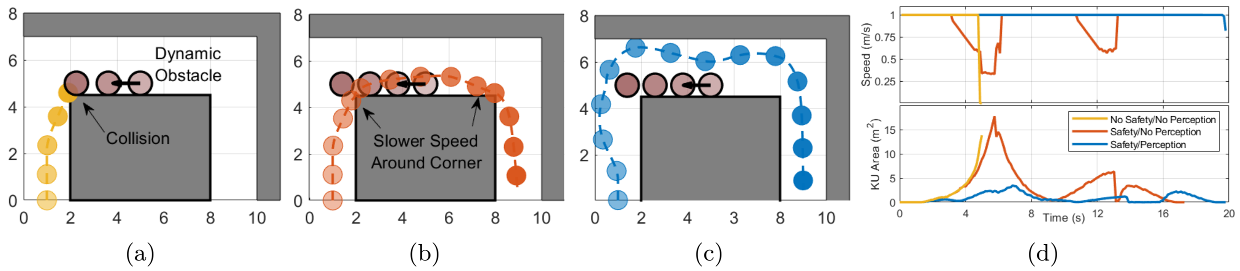


Figure 3.7: Simulation case study for a robot navigating occluding corners in the presence of a dynamic obstacle. The different color gradient in (a), (b) and (c) represent different time instances: lighter (darker) colors occur earlier (later) in the trajectory. The comparison between A_{ku} and speeds is presented in (d).

robot to not see the moving obstacle until it is too late and a collision occurs. Fig. 3.7(b) shows motion with only the safety module. The robot slows its velocity as it approaches the occluding corner, preventing a collision with the dynamic obstacle. Fig. 3.7(c) shows the policy that uses both the safety module and the perception objective inside the OCP of the MPC. This policy commands the robot to move around the occluding corner to improve perception. Not only does this prevent collision, but this also allows the robot to maintain the same speed and increase visibility through the operation. Fig. 3.7(d) compares the speed and A_{ku} over time for each of these runs, showing how the full control policy reduces the known-unknown area while maintaining a higher velocity.

To demonstrate how this policy also works in unstructured environments, Fig. 3.8 shows example motion through an unknown randomly generated dense forest, both with and without the perception objective. Fig. 3.8(a) shows the difference in trajectories that the perception objective makes. Without perception, the MPC commands motion close to obstacle boundaries, and with perception, the robot chooses a path often midway between trees. Fig. 3.8(b) compares the speed and known-unknown area over time for both. Our full approach not only reduces the known-unknown area while moving through the forest, but also results in movement at max speed for 82% of the trajectory, compared to 58% of the trajectory without perception.

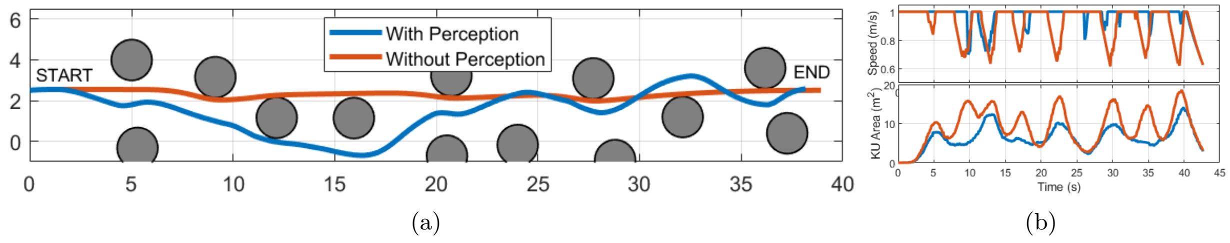


Figure 3.8: Simulation case study for a robot navigating a randomly generated forest.

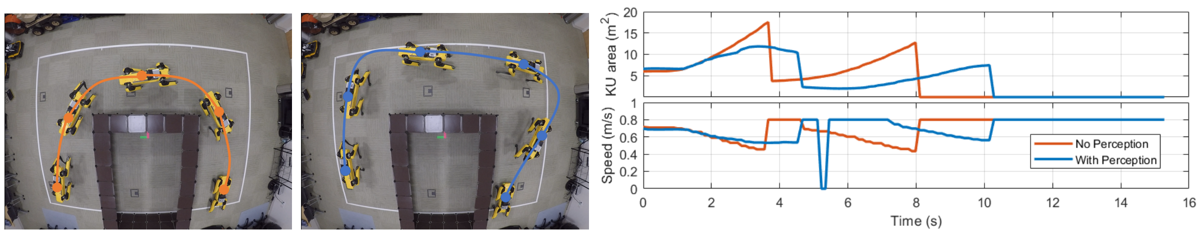
3.5 Experiments

The proposed approach was additionally verified on a Boston Dynamics' Spot platform, a quadrupedal robot that offers unique mobility capabilities that can outperform wheeled robots in more challenging terrain [11]. Spot can also be commanded using the same velocity motion model used in simulation, and so is an ideal use case for our approach. The OCP in (3.7) was solved at runtime in a hallway environment with two corners, similar to the simulation shown in Fig. 3.7. The prediction horizon and discretization time were set to be $T = 5$ s and $dt = 0.25$ s, respectively.

Fig. 3.9 shows the results of this experiment. Fig. 3.9(a) shows motion using only the safety module, with Spot slowing down as it cuts each corner around the hallway. Fig. 3.9(b) shows motion with our full approach that uses the perception objective to minimize occlusions in the environment. Fig. 3.9(c) compares the known-unknown area and speed in each example. Again, the perception objective not only reduces the known-unknown area around each occlusion, but also allows for faster motion. For motion with the perception objective, speed drops to 0.0 m/s only momentarily as Spot turns sharply around the first corner – otherwise, it maintains a near-max speed throughout the rest of the trajectory.

3.6 Discussion and Conclusion

This chapter presented a framework for visibility-based occlusion-aware navigation in unknown and unstructured environments. This framework built upon Chapter 2 in that it formulated an analytically simple perception objective that approximated the known-unknown area behind occluding obstacles and used this perception objective within the cost function of an OCP. This chapter improved upon these concepts by providing a perception objective that was defined in terms of circular occluding obstacles, which allowed it to be applied towards more unstructured environments. Additionally, a safety module was included that commanded safe speeds in unknown environments, enabling the robot to safely brake to a stop if necessary. Both visibility and safety



(a) Motion with safety module only. (b) Motion with both safety and perception. (c) Comparison of A_{ku} and speed, with/without perception objective.

Figure 3.9: Experiments and results for a hallway scenario with Spot.

were combined into the same framework, and it was shown through simulation and experiment how increasing visibility allowed the robot to maintain a maximum speed throughout the unknown environment.

As with Chapter 2, the analytically simple form of the perception objective allows the approach discussed here to be applied to an existing navigation stack with relatively little overhead and without any pre-training. Similarly, the resulting occlusion-aware behavior can heavily depend upon the relative weights of the different components within the cost function (3.4), meaning tuning may be required when deploying the proposed framework in various environments. Additionally, the frameworks presented in Chapter 2 and this chapter increase visibility around occlusions by design, and it is demonstrated through simulation and experiment how these frameworks are beneficial to the overall motion of the robot. Often times, increasing visibility around occlusions helps decrease the risk of poor motion created by dynamic obstacles behind these occlusions. Motivated by this observation, the remainder of this dissertation explores the concept of *risk-aware* motion planning as a means of approaching the occlusion-aware navigation problem. In this way, motion that increases visibility around occlusions may be considered an emergent property of a risk-aware policy, rather than assumed to be beneficial to the robot as in a visibility-based policy.

Part II

Data-Driven, Risk-aware Motion Planning

Chapter 4

A Model Predictive Path Integral Method for Fast, Proactive, and Uncertainty-Aware UAV Planning in Cluttered Environments

4.1 Introduction

This chapter presents an exploration into a risk-aware navigation policy that minimizes some risk metric over a planned trajectory. Specifically, this chapter deals with the risk of collision with static obstacles in the environment. Additionally, this risk metric is modeled as a machine-learned component, trained via example test flight data. Although the problem addressed in this chapter does not directly relate to occlusion-aware navigation, the concept of data-efficient learning and risk-aware planning apply to both Chapter 5 and 6, which treat the occlusion-aware navigation problem as a risk-aware navigation problem, efficiently using data to train a risk-sensitive policy.

Concretely, this chapter addresses the risk of collision due to trajectory tracking error from a low-level controller. Consider the scenario depicted in Fig. 4.1 in which an aerial robot must pass through a narrow opening to the other side. A receding-horizon motion planner may command a fast-moving trajectory to reduce travel time, but such a trajectory may not be perfectly tracked by the low level controller. This could induce a large tracking error, meaning that tracking a collision-free trajectory may not result in collision-free motion. In order to mitigate the risk of collision under this kind of uncertainty, the robot must command slower motion through the gap, reducing the tracking error.

One possible solution is a data-informed approach, where the tracking error and subsequent risk of collision are inferred from past performance of the robot. This data-informed risk assessment allows the risk measure to accurately reflect the performance of the low level controller, but must capture a potentially complex relationship between the commanded trajectory and the risk of tracking that trajectory. Gradient-based and quadratic-programming-based approaches are restrictive in that risk-

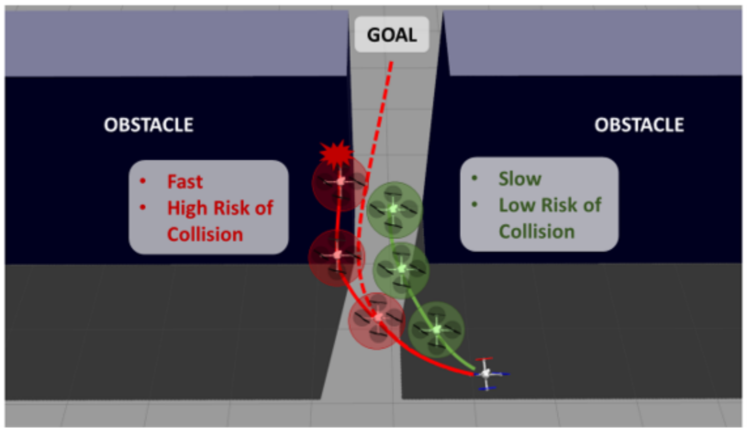


Figure 4.1: Motivating example in which a slower speed results in safer motion through a small gap.

based costs must have certain numerical qualities for real-time use. Alternatively, sampling-based approaches consider costs with minimal assumptions, allowing a greater flexibility and generality when defining risk. For this reason, the heart of the proposed approach in this chapter is a receding-horizon Model Predictive Path Integral (MPPI) motion planner, adapted from the sampling-based MPPI control used in information theoretic control theory [88]. Typical MPPI works by rapidly sampling the low level control space of the system around a “best guess” of the optimal open-loop control policy, and a weighted sum is performed to iteratively update this best guess, converging to the optimal control policy after many iterations. One consequence of sampling within the low level control is the need for a large number of samples at a high rate (typically on the order of 50-100 Hz). For real-time use, this requires the robot to have a GPU on board to speed up the sampling time.

Our approach adopts MPPI control to a trajectory planning setting; instead of sampling within the control space, we sample within a trajectory parameter space. In particular, the proposed MPPI trajectory planner determines waypoints that define a spline-based trajectory. This reduces the sampling space dimension, allowing our MPPI path planner to run on a CPU in real-time, hence for a more relaxed set of system requirements. Overall, our approach allows for a computationally more efficient method to sample trajectories within the MPPI framework, without sacrificing the ability to generate fast and safe trajectories.

This chapter presents two main contributions. First, the MPPI control approach is cast as a parameterized high level planner, reducing the dimension of the optimization space and mitigating the hardware requirements needed to find reasonable solutions to the motion planning problem. Second, a data-informed risk measurement is included inside the cost function of the MPPI motion planner, using the actual trajectory tracking performance of the system to determine safe and lively trajectories. The result is motion that minimizes risk of collision due to trajectory tracking error while avoiding obstacles and moving towards a goal. Together, the overall approach provides a practical method for run time risk-aware trajectory generation towards safe navigation. While this approach is flexible enough to be applied in a general motion planning setting, this chapter focuses on motion planning for an unmanned aerial vehicle (UAV) since these types of systems can be greatly affected by tracking error, and such tracking error can induce risk of collision, especially when navigating cluttered environments.

4.2 Problem Formulation

In this work, we are interested in creating a receding-horizon trajectory generation policy that addresses the risk of collision due to tracking error between the commanded trajectory and the actual trajectory of a UAV, especially when navigating potentially cluttered environments. Additionally, this trajectory generation policy should be able to handle data-informed functions of risk, and consider potentially complex relationships by placing minimal assumptions on the properties such functions may have (e.g. smoothness, differentiability). We separate this problem into two parts: (i) creation of a receding-horizon trajectory generator that can optimize for a general cost function at run time, and (ii) the inclusion of a risk factor that, when minimized, commands safe and lively trajectories in the presence of low level tracking error.

Problem 1: *Receding Horizon Trajectory Generation*: We seek a policy $\mathcal{P}_\tau(\boldsymbol{x}(t_0))$ that takes in the current state of the robot $\boldsymbol{x}(t_0) \in \mathbb{R}^{n_x}$ and at run time returns a time-based trajectory $\tau(t)$ defined over a future horizon $t \in [t_0, t_0 + t_H]$ for a low level controller to track. This trajectory should move the robot closer to a goal state $\boldsymbol{x}_g \in \mathbb{R}^{n_x}$, as well as avoid the state set $\mathcal{X}_O \in \mathbb{R}^{n_x}$ occupied by obstacles:

$$\begin{gathered} |\boldsymbol{x}(t_0 + t_H) - \boldsymbol{x}_g| \leq |\boldsymbol{x}(t_0) - \boldsymbol{x}_g| \\ \boldsymbol{x}(t) \notin \mathcal{X}_O, \, \forall t' \in [t_0, t_0 + t_H] \end{gathered} \tag{4.1}$$

In addition to these requirements, this policy should optimize over a cost function $S(\tau)$ that may be nonlinear, non-smooth and even non-differentiable:

$$\mathcal{P}_\tau = \arg\min_\tau \left[S(\tau)\right] \tag{4.2}$$

Note that the commanded trajectory $\tau(t)$ may not be the same as the actual trajectory τ_{act} of the system over time, due to tracking error. To compensate for this, the proposed approach also considers a risk measure $\rho(\cdot) \in \mathbb{R}$ that relates a given trajectory $\tau(t)$ to the risk of collision due to this error $|\tau(t) - \tau_{\text{act}}(t)|$. Although we do not constrain ρ to have any particular form, basic assumptions must be placed in order to cast the problem of risk minimization correctly: (i) $\rho(\cdot)$ is positive semi-definite, (ii) $\rho(\cdot) = 0$ for situations that have no risk, and (iii) $\rho(\cdot) > 0$ for situations that have risk of collision. With these assumptions, this risk can be included inside the cost function of the trajectory generation policy.

Problem 2: *Risk-aware Navigation*: Given a risk measure $\rho(\cdot)$, create a policy $\mathcal{P}_\tau$ for finding a trajectory τ that also minimizes risk:

$$\mathcal{P}_\tau = \arg\min_{\tau} \left[S(\tau) + \int_\tau \rho(\cdot) dt \right] \tag{4.3}$$

Next, we discuss our proposed approach for safe, risk-aware navigation of a UAV by combining risk measures with a novel MPPI-based motion planning technique.

4.3 Approach

Fig. 4.2 shows a diagram of the proposed approach. Trajectories $\tau(t, R)$ are parameterized with respect to both time t and R. The current robot and environment state are fed into the MPPI trajectory planner, which has a generalized risk function $\rho(\cdot)$ that characterizes the risk of a particular trajectory. The output of this trajectory planner is a set of parameters R^* that optimizes over a general cost function $S(\tau)$ that includes go-to-goal and obstacle avoidance objectives, as well as this risk of collision:

$$R^* = \arg\min_{R} \left[S(\tau) + \int_\tau \rho(\cdot) dt \right] \tag{4.4}$$

As a concrete running example for this work, risk-aware navigation of a quadrotor was chosen as a straight-forward yet elucidating system. In particular, the MPPI trajectory planner sought to minimize risk of collision due to trajectory tracking error; that is, for a given trajectory determined by parameters R, the risk associated with a lower-level controller not perfectly tracking this error and colliding with an obstacle.

4.3.1 Risk Measure Formulation

This section proposes a data-informed risk measure that models geometric mismatch between the trajectory $\tau(t, R)$ tracked by the low level controller and the actual motion of the UAV, τ_{act}.

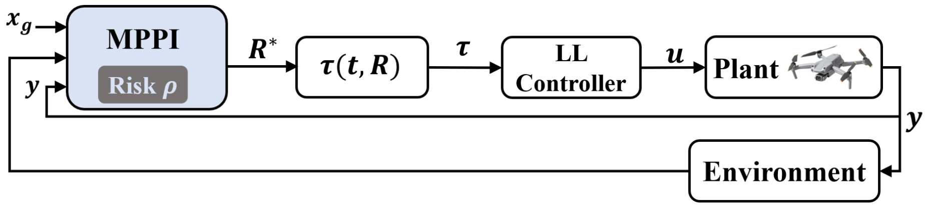


Figure 4.2: Diagram showing the proposed motion planner (blue shaded cell) within the context of a general autonomy stack.

Specifically, a relationship is established between this mismatch and the maximum speed commanded by $\tau(t, R)$. This relationship captures how higher robot speeds often worsen tracking error of a desired trajectory by the low level controller. This degradation in performance can lead to unsafe situations, especially when the robot is travelling in a cluttered environment. Thus, the risk measure $\rho(\tau(t, R))$ relates the speed of the commanded trajectory to the risk of collision with nearby obstacles.

In order to define this risk measure, first define $d(t_1, t_2)$ as the euclidean distance between a point $\tau_1(t_1)$ on one trajectory and point $\tau_2(t_2)$ on another. The Hausdorff distance $d_H(\cdot)$ between two trajectories τ_1 and τ_2 is defined as:

$$d_H(\tau_1, \tau_2) = \max\left\{ \max_{t_1 \in [0,PT]} \left[\min_{t_2 \in [0,PT]} d(t_1, t_2) \right], \max_{t_2 \in [0,PT]} \left[\min_{t_1 \in [0,PT]} d(t_2, t_1) \right] \right\} \tag{4.5}$$

Fig. 4.3 shows an example of how $d_H(\cdot)$ is found between the commanded trajectory $\tau(t, R)$ and the actual trajectory τ_{act} that results from trying to track $\tau(t, R)$.

If $d_H(\tau(t, R), \tau_{\text{act}}) \approx 0$, then both trajectories have considerable overlap in the xyz plane over the entire trajectory, while $d_H(\tau(t, R), \tau_{\text{act}}) \gg 0$ signifies at least some portion where there is significant deviation. Through simulation or experiment, data can be collected that measures $d_H(\tau(t, R), \tau_{\text{act}})$ for various commanded trajectories. Specifically these data can be used to create an estimator of the deviation as a function of some states and inputs. For the specific UAV application considered in this chapter we have observed – as intuitively expected – that the deviation is a function of the maximum speed v_{max} commanded by $\tau(t, R)$. In this way it is possible to predict the tracking error using only information from the commanded trajectory.

Denote d_{obs} as the distance between $\tau(t, R)$ and the nearest obstacle. For the UAV, it is considered risky when $d_{\text{obs}} < d_H$, since the deviation of the actual trajectory may extend toward the obstacle, potentially colliding with it. Likewise, if $d_{\text{obs}} \geq d_H$, then there is no risk of collision, since the robot is expected to deviate from $\tau(t, R)$ by a distance smaller than the nearest obstacle.

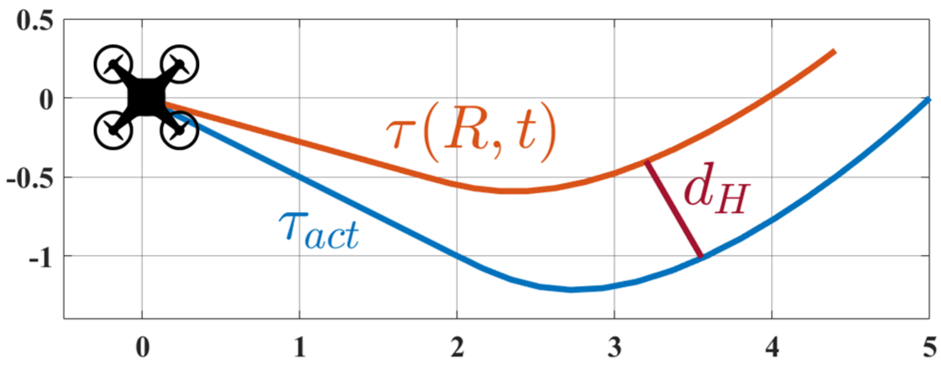


Figure 4.3: Example of trajectory tracking error.

To this end, the risk measure is defined as:

$$\rho(\tau(t,R)) = \max\left[0, \frac{d_H}{d_{\text{obs}}} - 1\right] \tag{4.6}$$

In this way, $\rho(\cdot) > 0$ when there exists the potential for τ_{act} to intersect with the boundary of an obstacle, and $\rho(\cdot) = 0$ when $d_{\text{obs}} \geq d_H$.

4.3.2 MPPI for Motion Planning

MPPI control is a sampling-based control method to find the solution to a stochastic optimal control problem (OCP). In the proposed approach, the MPPI solver is used to find a series of waypoint positions $R = \{\boldsymbol{r}_1, \boldsymbol{r}_2\}$ that define a trajectory $\tau(t, R)$.

The MPPI algorithm must be defined with respect to some stochasic equations of motion for the state $\boldsymbol{x}$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \tau(t_k, R + \mathcal{E})) \tag{4.7}$$

Here, $\boldsymbol{f}(\cdot)$ represents a discrete-time equation of motion in which the robot $\boldsymbol{x}$ evolves under the influence of a trajectory $\tau(t, R+\mathcal{E})$, where $\mathcal{E} = \{\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2\}$ are random perturbations that shift the p^{th} waypoint by $\boldsymbol{\epsilon}_p \sim \mathcal{N}(0, \Sigma)$.

The stochastic optimal control problem may be defined as the minimization of an expectation value, denoted by $\mathbb{E}(\cdot)$:

$$R^* = \arg\min_{R} \mathbb{E}_{\mathbb{Q}}\left[S(R+\mathcal{E})\right] \tag{4.8}$$

The term $S(\cdot)$ defines the cost of the total trajectory, with waypoints R being perturbed by stochastic variables $\mathcal{E}$ that create the probability distribution $\mathbb{Q}$.

The total cost $S(\cdot)$ is defined over $P = 2$ waypoints as:

$$S(R+\mathcal{E}) = \phi(\boldsymbol{x}(PT)) + \int_0^{PT} \mathcal{C}(\boldsymbol{x}(t))dt \tag{4.9}$$

The term $\phi(\cdot)$ is a terminal cost function defined as the error between the final state $\boldsymbol{x}(PT)$ and the goal state:

$$\phi(x(PT)) = w_g|\boldsymbol{x}(PT) - \boldsymbol{x}_g| \tag{4.10}$$

The constant $w_g > 0$ is a scaling factor that is tuned to adjust the relative weight of the different objectives within (4.9). The running cost $\mathcal{C}(\boldsymbol{x}(t))$ is defined by two terms:

$$\mathcal{C}(\boldsymbol{x}) = \mathcal{C}_{obs}(\boldsymbol{x}) + \mathcal{C}_{\rho}(\boldsymbol{x}) \tag{4.11}$$

The first term $\mathcal{C}_{obs}$ is an obstacle cost that heavily penalizes collisions with known obstacles in the environment. Since MPPI is a gradient-free method, the exact form of $\mathcal{C}_{obs}$ can be quite sparse with gradient information, such as an indicator function used with an occupancy map. For the obstacle course assumed by this chapter, the obstacle cost is defined using an indicator function $I_{o_j}(x)$ that returns 1 when state $\boldsymbol{x}$ lies within obstacle o_j, else it returns zero.

$$\mathcal{C}_{obs} = w_{obs} \sum_{j=0}^{n_o} \int_0^{PT} I_{o_j}(\boldsymbol{x}(t))dt \tag{4.12}$$

The second term $\mathcal{C}_\rho$ is the portion of the cost related to the risk of a trajectory. Since we constrain $\rho(\cdot)$ to be positive semi-definite, the risk cost can be defined as proportional to this risk measure:

$$\mathcal{C}_\rho = w_\rho \rho(\cdot) \tag{4.13}$$

As was the case with the obstacle cost, using an MPPI-based approach to solving (4.8) allows the exact form of $\rho(\cdot)$ to be quite flexible in its definition, since it does not require information about the gradient of $\rho(\cdot)$. For example, $\rho(\cdot)$ may be a function that approximates some notion of risk that may be hard to write by hand, e.g., a learned policy trained on simulated or experimental data.

Inclusion of the risk measure defined by (4.6) in the cost function has two different effects. First, trajectories are generally planned to be spatially away from obstacles in order to increase d_{obs}. Second, if the robot must move through small gaps in order to reach its goal, then trajectories that command slower motion are preferred in order to decrease d_H.

Authors in [89] show how it is possible to obtain a theoretical *exact* solution to (4.8). Unfortunately it is impossible to compute directly, but may be approximated using an iterative sampling method. This iterative algorithm relates the $(k+1)^{\text{th}}$ iteration to the k^{th} iteration by:

$$R_{k+1} = R_k + \sum_{i=1}^{N} w(\mathcal{E}_i)\mathcal{E}_i \tag{4.14}$$

$$w(\mathcal{E}_i) = \frac{1}{\eta} \exp\left[-S(R_k + \mathcal{E}_i)\right] \tag{4.15}$$

Here, η is a normalization factor to ensure $\sum_i^N w(\mathcal{E}_i) = 1$. Fig. 4.4 illustrates how this algorithm finds a trajectory around an obstacle with $P = 2$ waypoints. At the k^{th} iteration, the current waypoint locations R_k are subjected to a series of random perturbations $\{\mathcal{E}_i\}$. The blue/green trajectories are the results of these perturbations, with the color corresponding to the weight of the trajectory as determined by (4.15). The $(k+1)^{\text{th}}$ iteration is found through a weighted sum of these perturbations, so that $\tau(t, R_{k+1})$ has a lower cost than the previous iteration. This procedure is

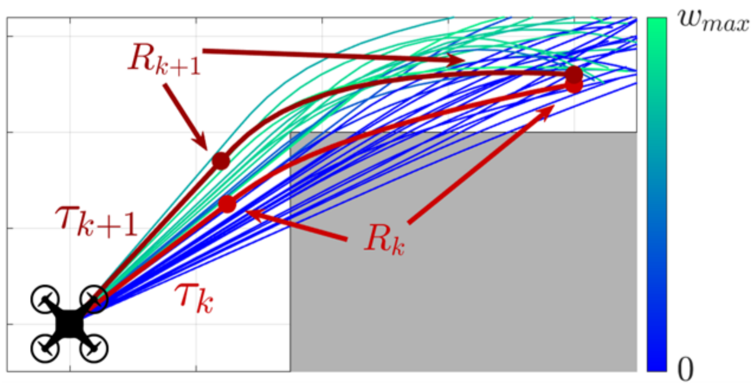


Figure 4.4: An example iteration step of the MPPI algorithm.

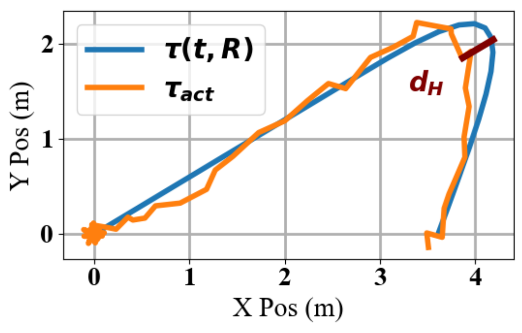
repeated n_{iter} times, and the resulting waypoints $R_{n_{\text{iter}}}$ are applied. In general, n_{iter} is chosen to be as large as possible so that the iterative procedure can reasonably converge to the optimal solution.

4.4 Simulations

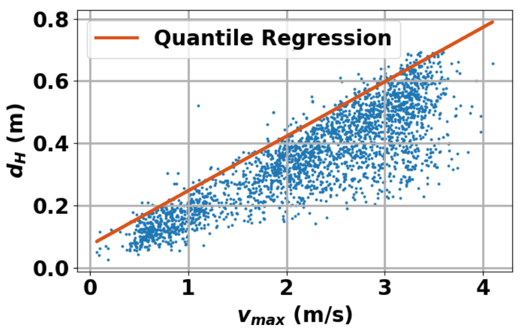
Simulations were performed to validate the ability of the proposed approach to reduce risk and negotiate obstacles in a cluttered environment while navigating towards a goal. RotorS [34] was used as a high-fidelity simulator for UAV motion that also includes a low level trajectory nonlinear controller [50]. Given a commanded trajectory $\tau(t)$, the low level controller attempts to track this trajectory, but due to measurement noise and physical limitations the actual trajectory τ_{act} is somewhat different, leading to a non-zero tracking error $d_H(\tau(t), \tau_{\text{act}}(t))$. In order to estimate $d_H(\cdot)$, data were collected by commanding different trajectories $\tau(t)$ and recording the resulting trajectory τ_{act}. Each trajectory was defined by $P = 2$ waypoints placed in the xyz plane, with $T = 2.5$ seconds between each waypoints. Noise was injected into the simulated odometry sensor as a way to exacerbate the tracking error of the low level controller.

Fig. 4.5(a) shows a recorded example in which τ_{act} differs from $\tau(t)$. Also included in this plot is $d_H(\cdot)$ between the two trajectories. Fig. 4.5(b) shows a plot of $d_H(\cdot)$ as a function of maximum speed $v_{\max}$ along $\tau(t)$. It can be seen that as $v_{\max}$ increases, the set difference $d_H(\cdot)$ also increases. A regression line $\hat{d}_H(v_{max})$ was fit to the data to capture the 95th percentile, giving a conservative estimate of the actual set difference, $\hat{d}_H(v_{max}) \approx d_H(\cdot)$. This estimator of the set difference $\hat{d}_H$ was used in (4.6) to find an estimate of the risk $\rho(\cdot)$.

The MPPI algorithm that solves (4.8) was written in C++, using perturbation covariance matrix $\Sigma = diag(0.15, 0.15, 0.0)$ m, $N = 50$ samples per iteration and $n_{\text{iter}} = 200$ total iterations per MPPI sample. The MPPI was run on a Lenovo ThinkPad X1 with Intel i7 6-core processor, and took an average of 0.19 ± 0.03s to run. The algorithm took the current state of the UAV $\boldsymbol{x}_0$ as well as a set of local obstacles $\{\boldsymbol{o}_j\}$ and returned an optimal set of waypoints R^* that defined a trajectory



(a)

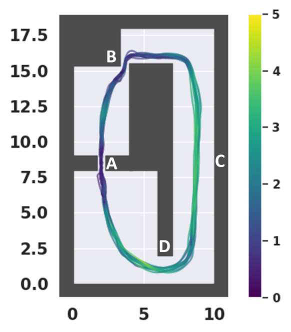


(b)

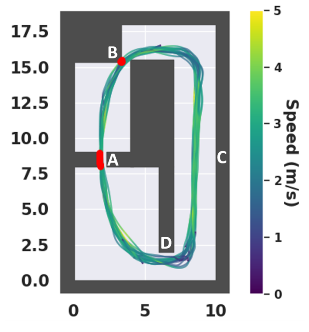
Figure 4.5: Correlation between perception objective $\Lambda(\cdot)$ and true known-unknown area A_{ku}.

$\tau(t, R^*)$ to be tracked. This trajectory was re-planned in a receding-horizon fashion, with the MPPI being re-sampled at a rate of 1 Hz to find an updated set of waypoints.

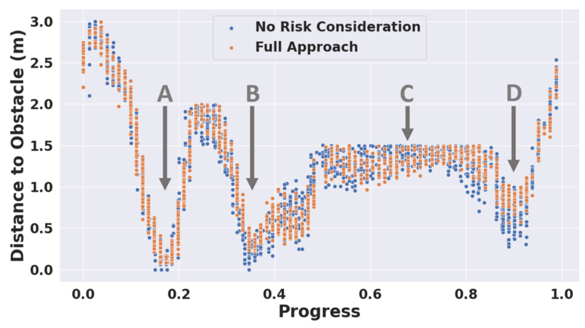
To test the capabilities of the MPPI planner, the UAV was tasked with navigating an obstacle course, shown in Fig. 4.6, with both small and large gaps to negotiate. To facilitate clockwise motion around the track, goal points were chosen a priori and commanded sequentially as the goal state in the MPPI cost function (4.10). These goal states also acted as a warm-start for the MPPI algorithm, choosing the initial waypoints R_0 to be along these goal points.



(a)



(b)



(c)

Figure 4.6: Simulation of quadrotor navigating an obstacle course using the MPPI motion planner.

Fig. 4.6(a) visualizes the resulting motion of the UAV as it tracked trajectories commanded by the MPPI. To highlight the effect of the risk objective on the overall motion of the UAV, Fig. 4.6(b) shows the resulting trajectories with $w_\rho = 0$ in the risk cost objective (4.13). This effectively removed the consideration of risk within the MPPI when planning motion, instead finding a trajectory that avoided obstacles while moving as quickly as possible. Qualitatively, the difference between these trajectories is only apparent when the UAV approaches small gaps **A** and **B**. With our full risk-aware approach, the UAV slowed down enough to ensure safe passage through these tight spots, whereas the policy with no risk consideration sped through these gaps, resulting in collisions due to tracking error. These collisions are shown visually by the red dots in Fig. 4.6(b). Alternatively, the risk-aware approach commanded the same high speed as the risk-agnostic policy through corridor **C**, since there were no close obstacles and it was safe to move quickly through this region.

Fig. 4.6(c) additionally shows the distance between the UAV and the nearest obstacle as it traveled around the track. This distance is plotted against progress along the track, where progress = 0 when the UAV was at the start of the course, and progress = 1 at the end of the course. This plot shows how the full approach worked to increase the distance between the UAV and obstacles, whereas the MPPI without risk consideration commanded motion closer to obstacles. Over the 20 laps, the full approach with risk consideration had 0 collisions, while motion with no consideration for risk resulted in 4 collisions.

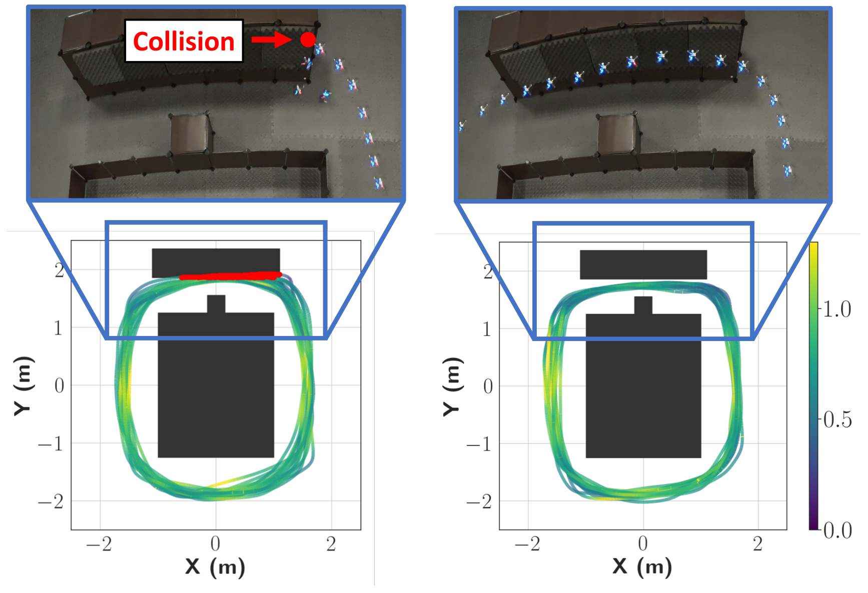
4.5 Experiments

The proposed approach was additionally verified experimentally on a Bitcraze Crazyflie quadrotor in a rectangular loop case study. A Vicon motion capture system provided odometry information to an offboard laptop, which then used the MPPI path planner with the same parameters as in simulation to send trajectories to the Crazyflie's non-linear controller [72]. The effect of including the risk measure inside the MPPI cost function is shown by comparing the full approach to the same MPPI with $w_\rho = 0$ in (4.13).

4.5.1 Rectangular Loop Case Study

The Crazyflie was tasked to complete loops around a central rectangular obstacle while negotiating its way through a narrow 30 cm gap between the northern wall and a protruding square obstacle. The top portion of Fig. 4.7 shows a snapshot of a sample pass through the narrow gap, both without risk consideration and with our full approach. For these single trajectory examples, the physical obstacle height was raised to demonstrate a collision without risk consideration. For the rest of data collected, the physical obstacle height was lowered to allow multiple laps around the environment without disruption.

The results of the multiple laps are shown in the bottom plots of Fig. 4.7. For this study, 20 laps were recorded for both the no risk and full approach cases, giving 40 total laps tested. Fig. 4.7(a) shows how the UAV collided with the north wall 6 times (highlighted in red) when risk was not taken into account due to overshooting the planned MPPI trajectories at high speeds. However, with the full approach (Fig. 4.7(b)), no collisions occurred because the UAV slowed down at the bends of the loop in order to mitigate the overshooting behavior. Additionally, the UAV achieved comparable speeds both without risk consideration and with our full approach on the east, south and west side of the central square. This demonstrates how our full approach proactively adapted



(a) (b)

Figure 4.7: Example situation in which the safety module is used to command a slower velocity around an obstacle.

to move quickly through regions where there were no close obstacles, and the risk of collision due to tracking error was minimal.

4.5.2 4-Way City Block Case Study

In the second case study, the UAV was tasked to complete a complex path that involved alternating between straight lines and turning in different directions through a 4-way city block-like circuit. First, the UAV passed through a narrow 20 cm horizontal channel in the center of the configuration (Fig. 4.8(a)). It then executed a u-turn around the narrow rectangular obstacle in the second quadrant (Fig. 4.8(b)) and went through a 20 cm vertical passage (Fig. 4.8(c)) before looping back to the start (Fig. 4.8(d)). Fig. 4.9 shows the trajectory of the Crazyflie over 10 laps in both the no risk and full approach cases, giving 20 laps total. As can be seen from the results, in the no risk case (Fig. 4.9(a)), the UAV collided with obstacles on 8 occasions within the narrow corridors, while in the full approach (Fig. 4.9(b)) it never collided. The speed profiles also demonstrate that, when obstacles are far enough away, the full approach allowed the UAV to reach the same speeds as with no risk consideration. Furthermore, thanks to our risk-aware framework, the UAV slowed down when traversing the cluttered sections of the environment, allowing for safer navigation that avoided collisions.

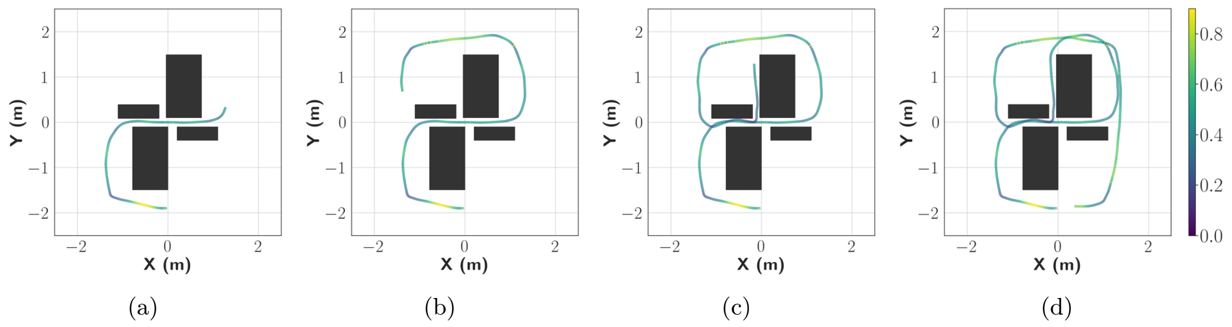


Figure 4.8: A demonstration of a single lap that the UAV performs around the 4-way city block environment, along with its velocity profile.

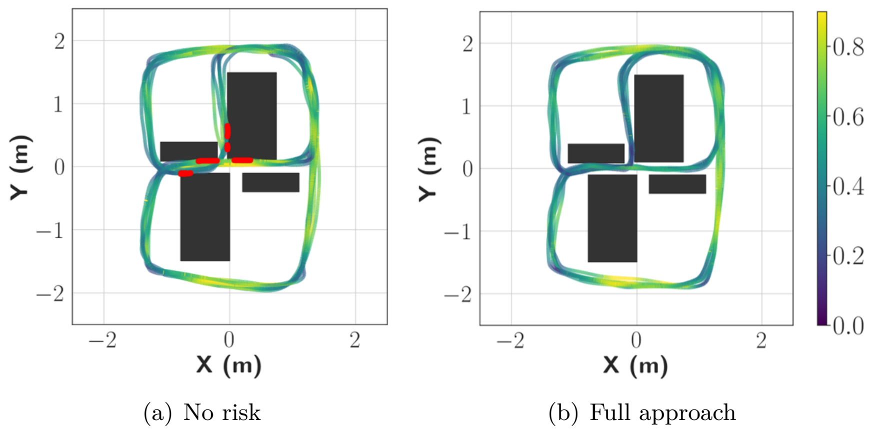


Figure 4.9: Crazyflie positions and velocities for the 4-way city block environment (a) without considering risk and (b) with the full approach.

4.6 Discussion and Conclusion

This chapter presented a receding-horizon path planning approach that can proactively adapt the trajectory of a robot at run time in order to reduce overall risk while navigating through a cluttered environment. The proposed approach utilizes an MPPI control algorithm in order to accommodate a general, data-informed risk measure. Importantly, the trajectory planned by the MPPI is parameterized by only a few number of variables, greatly reducing the computational requirements to run the MPPI algorithm and allowing the approach to run on more general hardware. The full approach was validated on a UAV robotic system navigating around obstacles towards a goal, with risk defined by the tracking error between commanded trajectory and the actual trajectory. Both simulation and experiment demonstrated how the inclusion of this risk measure inside the cost function allows the robot to move more safely through the environment, compared to motion without consideration for risk.

Despite this problem not being directly related to occlusion-aware navigation, Chapter 5 and 6 both apply the concepts developed in this chapter towards the creation of occlusion aware policies.

First, the concept of risk-aware motion planning is applied towards the problem of occlusion-aware navigation, and a metric is developed that quantifies the risk associated with occluded regions. Second, this risk metric is informed through data collected at run time, efficiently using such data to create a risk-sensitive policy at run time.

Part III

Data-Driven, Risk-based Occlusion-Aware Motion Planning

Chapter 5

Data-Driven Occlusion-Aware Navigation via Online Quantile Temporal Difference Learning

5.1 Introduction

Chapter 2 and Chapter 3 introduced a visibility-based approach towards occlusion-aware navigation in which an analytical function that approximated the geometric area behind occlusions was included as a term inside the cost function of an OCP. Doing so resulted in occlusion-aware motion that balanced between promoting visibility around occlusions and minimizing travel time to goal, and the overall effect was a greater reaction time to account for dynamic obstacles in the environment, which in turn resulted in both smoother and faster motion. One question that follows from this work is if a policy can instead *learn* how occlusions affect the experience of the ego robot, using data collected at run time as a means of learning how to avoid certain occlusions in the environment. Doing so may still produce motion that promotes visibility around occlusions, but such a behavior would be emergent as the policy learns to avoid negative experiences, instead of being assumed a priori.

To explore these concepts, this chapter proposes an approach for occlusion-aware navigation in dynamic environments, collecting data and learning how to improve visibility at run time. This is achieved through a classical planning algorithm outfitted with a global cost map that encodes the risk of experiencing a negative outcome. This risk cost map, or "risk map," is learned from data collected through run time interactions with the dynamic environment. Because of this, the learned risk map is bespoke to the unique factors that contribute to negative outcomes, such as the placement of the dynamic obstacles within the environment, as well as the movement patterns of these dynamic obstacles. This risk map is created using the Quantile Temporal Difference (QTD) [24] algorithm, a technique borrowed from distributional reinforcement learning, to determine the distribution of negative outcomes that result from dynamic obstacles being occluded within the environment. Creating this distribution imbues the risk map with a certain sensitivity towards infrequently encountered outcomes to achieve a risk-sensitive policy. Thus the classical planning algorithm serves as a starting point for the proposed approach, incrementally learning this risk map while simultaneously seeking to avoid areas of high risk.

Our work presents two main contributions. First, we propose the use of QTD learning to create a risk map, represented as a global cost map that encodes risky areas of occlusion-related negative outcomes for a global path planner to avoid. This particular representation is favorable because it allows the risk map to encode risk associated with individual occlusions within the environment (rather than considering all occlusions as the same), and its simplified input structure enables the risk map to be generated quickly using relatively small amounts of data. Second, we leverage these characteristics to create this risk map at run time, enabling the policy to adjust to new data as they are collected. The result is a path planning approach that may be immediately used out of the box, and can adjust its paths "on the fly" in order to reduce the experience of negative outcomes.

5.2 Problem Formulation

In this work, we are interested in creating a motion planning policy Π that can account for occlusions created by static obstacles in the environment. The proposed approach casts the problem of occlusion aware navigation as a *risk aware* navigation problem, in which the occlusions actively cause a risk of encountering a negative outcome, e.g., the robot is too close to a dynamic obstacle. This outcome is denoted as a binary variable $C \in \{0, 1\}$, with $C = 1$ denoting when the robot experiences a negative outcome. Although not strictly required, C is treated in this work as caused by the presence of occluded dynamic obstacles moving with an unknown policy. Thus, $C = 1$ serves to denote an avoid-set for the robot, one which is possibly entered from certain problematic states in the environment. A function $\rho(\boldsymbol{x})$ can be created from past experience that maps from robot state $\boldsymbol{x}$ to an associated risk of encountering $C = 1$.

Concretely, let $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ define the dynamics of a robot system with state $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and control signal $\boldsymbol{u} \in \mathbb{R}^{n_u}$, which is derived from a lower level controller $g(\cdot)$ attempting to track a path τ so that $\boldsymbol{u} = g(\tau, \boldsymbol{x})$. Define $\mathcal{X}$ as the set of possible robot states, with $\mathcal{X}_o \subset \mathcal{X}$ defined as the area occupied by static and dynamic obstacles in the environment.

Problem 1: ***Run Time Risk Mapping*****:** Let Π define a policy that generates τ from robot state $\boldsymbol{x}$ to goal set $\mathcal{X}_g$ while avoiding $\mathcal{X}_o$.

$$\tau = \Pi(\boldsymbol{x}, \mathcal{X}_g, \mathcal{X}_o, \cdot). \tag{5.1}$$

Furthermore, define $C_t \in \{0, 1\}$ as a stochastic random variable that denotes the experience $C_t = 1$ or non-experience $C_t = 0$ of a negative outcome for the robot at time t. We seek a risk measure

$\rho : \mathbb{R}^{n_x} \to \mathbb{R}$ that maps from the robot state $\boldsymbol{x}$ to a positive semi-definite scalar ρ that describes the risk of encountering a negative outcome from using policy Π. This function $\rho(\cdot)$ should be trained from statistics discerned by data gathered at run time and collected into a replay buffer $\mathcal{R}$.

Typically, the policy Π is defined to minimize some overall cost S_0, such as distance traveled. However, learning this risk map ρ at run time provides an additional opportunity for the path planning policy Π to be adjusted as well. That is, the policy Π that generates data for the risk map ρ can be adjusted at run time to avoid risky states the next time the robot negotiates the same portion of the environment. In this way, the policy Π can actively avoid risks learned at run time.

Problem 2: ***Run Time Risk-Aware Adaptation*****:** We seek a pipeline that allows the motion planning policy Π to incorporate the risk map $\rho(\cdot)$ at run time,

$$\tau = \Pi(\rho(\cdot), \cdot). \tag{5.2}$$

Specifically, we seek a policy Π that finds a path τ to minimize the following cost function S,

$$S = \int_\tau \mathcal{L}_0(\boldsymbol{x}) + \lambda\rho(\boldsymbol{x})d\tau, \tag{5.3}$$

where $\mathcal{L}_0$ is the Lagrangian that results in occlusion-agnostic motion, so that $S_0 = \int_\tau \mathcal{L}_0 d\tau$. The weight $\lambda \geq 0$ defines the relative importance of avoiding risk based on past experience against following the original policy. In this way, Π may undergo policy improvement, actively avoiding risky situations.

5.3 Approach

Rather than creating a path planning policy entirely from scratch, we propose an approach that leverages a pre-existing policy Π_0, commonly available to many robots deployed in the real-world, that serves as an occlusion "unaware" policy that can effectively negotiate a known static environment.

Overtime, the current policy Π_k is evaluated to determine states in which negative outcomes $(C = 1)$ are experienced due to occlusions, and subsequently adjusted so that the new policy Π_{k+1} reduces the risk of a negative outcome by improving visibility. These steps of policy evaluation and policy improvement are commonly found within RL approaches, but are adjusted here to (i) instead encode the risk of entering an undesired set and (ii) be data-efficient so that this process can happen at run time. The run time risk-aware mapping and adaptation steps described in Sec. 5.2 thus refine the policy Π to account for negative interactions with dynamic obstacles due to occlusions. This allows our approach to plan motion with no initial training, and adapt to its environment over time as the robot gains more experience within this environment.

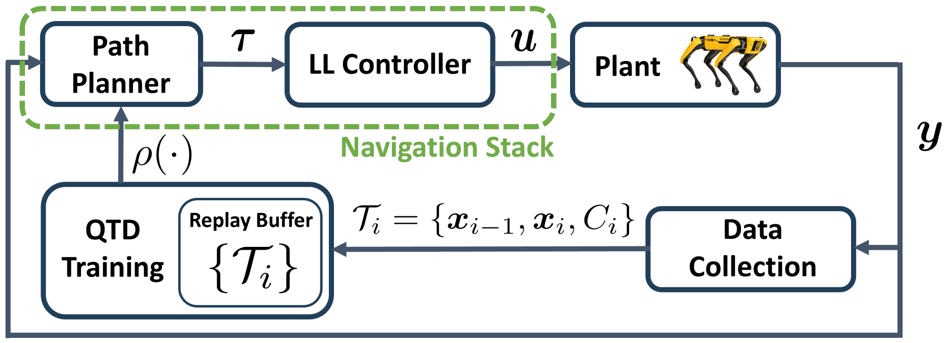


Figure 5.1: Diagram showing the proposed approach.

Fig. 5.1 shows a diagram of our approach. As the robot moves around and interacts with the environment, a data collection module tracks the previous state $\boldsymbol{x}_{i-1}$ and current state $\boldsymbol{x}_i$, along with the current outcome flag C_i, and sends this as training tuple $\mathcal{T}_i = \{\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, C_i\}$ to a QTD-based policy training module at run time. This module saves this data within a replay buffer $\mathcal{R} = \{\mathcal{T}_i\}$, using the replay buffer to continually train and update the risk map $\rho(\cdot)$ with this new data. Finally, this updated risk map is sent to the path planner module, placing a high-cost on risky states and avoiding them the next time the same occlusion is encountered. The path planner creates a trajectory τ that is tracked by a lower-level controller to produce a control signal $\boldsymbol{u}$.

The rest of Sec. 5.3 further details the proposed approach, where Sec. 5.3.1 discusses how outcome C_i is defined in any given situation, Sec. 5.3.2 details how this risk metric ρ is constructed, and Sec. 5.3.3 describes how these concepts are used within the proposed approach for run time learning.

5.3.1 Negative Outcomes

Negative outcomes $C = 1$ are a general way of defining what should be avoided by the robot, if possible. Because the proposed approach seeks to learn at run time, the robot should have the ability to determine from its current state $\boldsymbol{x}_t$ if it is experiencing a negative outcome, $C_t = 1$, or it is not, $C_t = 0$. Additionally, the robot can also track multiple negative outcomes that may be possible, and set $C_t = 1$ if any one is currently being satisfied. For the proposed approach, one sensible negative outcome is if the robot position $\boldsymbol{p}_t = [p_x(t), p_y(t)]^\intercal$ is inside a desired avoid radius r_{avd} for an obstacle with position $\boldsymbol{o} = [o_x, o_y]^\intercal$ and radius r_o:

$$C_{t,\text{avd}} = \begin{cases} 1 & \text{if } |\boldsymbol{p}_t - \boldsymbol{o}| \leq r_{\text{avd}} \\ 0 & \text{else} \end{cases} \tag{5.4}$$

Typically, the avoid radius is chosen such that $r_{\text{avd}} > r_o$, providing additional padding between the robot and obstacle.

For occlusion-aware navigation, it is also desirable for newly visible dynamic obstacles to first be sensed by the robot from an appropriate distance away r_{vis}. This allows an appropriate time for this new information to be incorporated inside the path planning module, producing smoother motion. This also serves to benefit the path planning policy of the dynamic obstacle, which can first sense the robot from at least r_{vis} distance away as well and adjust its own planned path accordingly. Let $\{\boldsymbol{o}\}_{\text{vis},t-1}$ define the set of obstacles that is visible to the robot at the previous time step $t-1$. If an obstacle $\boldsymbol{o}$ is observed at the current time t, an occlusion-aware outcome C_{occ} can be defined as:

$$C_{t,\text{occ}} = \begin{cases} 1 & \text{if } \boldsymbol{o} \notin \{\boldsymbol{o}\}_{\text{vis},t-1} \text{ and } |\boldsymbol{p}_t - \boldsymbol{o}| \le r_{\text{vis}} \\ 0 & \text{else} \end{cases} \tag{5.5}$$

Because we make no assumptions on the motion policy of the dynamic obstacles, it is difficult to characterize an appropriate distance r_{vis} from first principles alone. Instead, the proposed approach leaves this as a tunable parameter, which may be adjusted based on the overall quality of the motion.

The union of both $C_{t,\text{avd}}$ and $C_{t,\text{occ}}$ form the set of negative outcomes:

$$C_t = C_{t,\text{avd}} \vee C_{t,\text{occ}} \tag{5.6}$$

5.3.2 Risk Map

Over time, data tuples $\mathcal{T}_i = \{\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, C_i\}$ are collected into a replay buffer $\mathcal{R} = \{\mathcal{T}_i\}$ in order to train the risk map $\rho(\cdot)$. To effectively manage memory, this replay buffer is implemented as a double-ended queue, in which new experience is added to the front and, once the replay buffer becomes a certain maximum size, pops old data from the back. The function $\rho(\cdot)$ should take in a robot state $\boldsymbol{x}$ and return a risk value associated with that state, based on the data available in $\mathcal{R}$. In order to define this risk, consider the scenario shown in Fig. 5.2 in which a robot must negotiate a corner that occludes the presence of an incoming dynamic obstacle. The initial policy, unaware of any risk, may choose to closely cut around the corner, seeing the dynamic obstacle at state $\boldsymbol{x}_t$. Because $|\boldsymbol{p} - \boldsymbol{o}| \le r_{\text{vis}}$, this state is associated with a negative outcome, $(\boldsymbol{x}_t, C_t = 1)$. Additionally, due to the inherently sequential nature of the path planning problem, states that immediately preceded $\boldsymbol{x}_t$ should also be considered when calculating risk, since they could lead to a negative outcome. This sequential nature is readily captured by defining a discounted sum of costs, or return

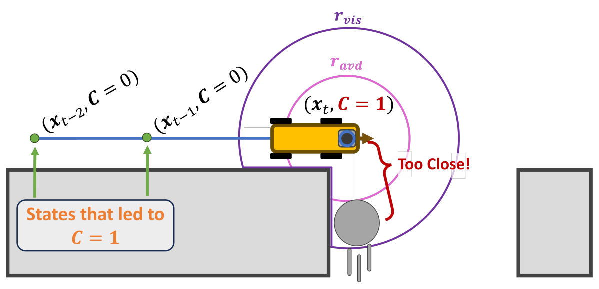


Figure 5.2: Due to the sequential nature of motion planning, states that preceded negative outcomes should also be considered when calculating risk.

cost G:

$$G_t = C_t + \gamma C_{t+1} + \gamma^2 C_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k C_{t+k} \tag{5.7}$$

Here, $\gamma \in [0, 1)$ is a discount factor that affects how G_t is influenced by future costs. The return cost defined in (5.7) is similar to the discounted sum of rewards usually defined for MDPs [80], the difference being the return cost should be minimized. This allows us to use techniques borrowed from reinforcement learning (RL) to leverage data of the form $\mathcal{T}_i = \{\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, C_i\}$ collected in the replay buffer $\mathcal{R}$ and create a risk measure ρ of encountering a negative outcome.

Typical RL techniques, such as temporal difference learning or proximal policy gradient, calculate the expected value of the return cost, which is often called risk-neutral because it is poor at tracking worst-case scenarios [53]. Since infrequent negative outcomes should also be accounted for, the proposed approach uses quantile temporal difference learning (QTD) [24] to learn the *distribution* of G_t from a given state. This distribution, denoted as $\eta^\Pi(\boldsymbol{x})$, depends on the path planning policy Π and the state of the robot $\boldsymbol{x}$, and is defined as

$$\eta^\Pi(\boldsymbol{x}) = \mathcal{D}_x^\Pi(G_t), \tag{5.8}$$

where $\mathcal{D}_x^\Pi$ is an operator that extracts the underlying probability distribution of the stochastic return cost G_t. Because probability distributions are infinitely dimensional, QTD uses a distribution representation of m equally spaced q-quantiles which are defined using parameters $\theta_i(\boldsymbol{x})$:

$$\{\theta_i(\boldsymbol{x}) \in \mathbb{R} : F_{\eta^\Pi(\boldsymbol{x})}(\theta_i(\boldsymbol{x})) = {}^{2i-1}/_{2m}\} \tag{5.9}$$

Here, $F_\eta(\cdot)$ is the cumulative distribution function of η, and $\theta_i(\boldsymbol{x})$ is the ${}^{2i-1}/_{2m}$-quantile of η. These q-quantiles are used to form an approximate distribution $\hat{\eta}^\Pi(\boldsymbol{x})$:

$$\hat{\eta}^\Pi(\boldsymbol{x}) = \sum_{i=1}^{m} \frac{1}{m} \delta_{\theta_i(\boldsymbol{x})} \tag{5.10}$$

where $\delta_{(\cdot)}$ is the dirac delta function that holds unit probability. QTD thus uses data from $\mathcal{R}$ to find quantiles $\theta_i(\boldsymbol{x})$ that accurately describes the return cost distribution. Further details may be found in [9].

Once these quantiles $\theta_i(\boldsymbol{x})$ are learned through QTD, they may be used to define a risk metric $\rho(\boldsymbol{x})$ for the return cost. In fact, the $^{2i-1}/_{2m}$-quantile is equivalent to the Value at Risk $\mathrm{VaR}_\alpha(G_t)$, with $\alpha = 1 - (^{2i-1}/_{2m})$, a common risk metric that can describe infrequent, worst-case scenarios [23]. To minimize computation and memory requirements, the number of quantiles m should reflect the desired α value to be tracked. For example, if $m = 32$, then the $^{2m-1}/_{m} = 98\%$-quantile is equivalent to $\mathrm{VaR}_{0.02}(G_t)$, defining the worst $\alpha = 2\%$ of outcomes for a particular state.

Thus, for a user-defined risk value α the proposed approach uses the learned $^{2m-1}/_{m}$-quantile, defined as $\theta_m(\boldsymbol{x})$, as the metric to associate a robot state $\boldsymbol{x}$ to a risk value ρ:

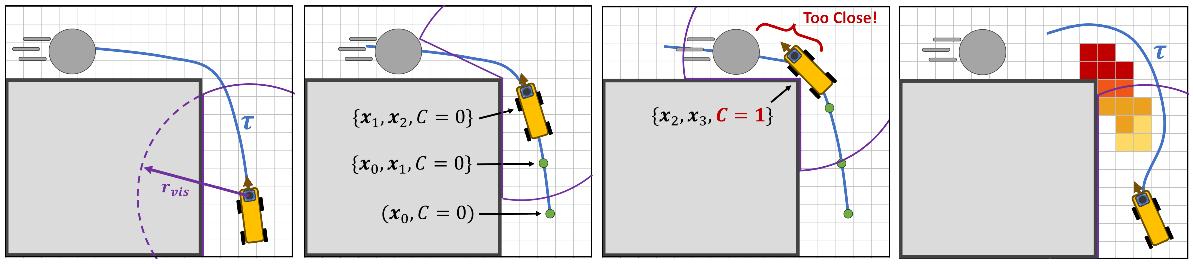
$$\rho(\boldsymbol{x}) = \theta_m(\boldsymbol{x}). \tag{5.11}$$

We close this section by noting that we consider $C = 1$ as a *terminal* outcome, meaning once $C = 1$ is encountered, the state trajectory is terminated. With this definition, it follows from (5.7) that the return cost $G_t \in [0, 1]$, since the worst-case return cost is an immediate experience of $C_t = 1$. Additionally, because the risk ρ is a quantile of the return cost G_t, this also bounds the risk map

$$0 \leq \rho(\cdot) \leq 1. \tag{5.12}$$

5.3.3 Motion Planning Policy and Run Time Learning

The process detailed in Sec. 5.3.2 is a technique for evaluating the risk of motion planning policy Π for particular states and creating a risk map $\rho(\boldsymbol{x})$. If done at run time, $\rho(\boldsymbol{x})$ can likewise be used by policy Π to avoid states that lead to $C = 1$ during the motion planning stage, which is essentially a policy improvement process. Together, both policy evaluation and improvement form the general


τ
r_{vis}
$\{\boldsymbol{x}_1, \boldsymbol{x}_2, C = 0\}$
$\{\boldsymbol{x}_0, \boldsymbol{x}_1, C = 0\}$
$(\boldsymbol{x}_0, C = 0)$
Too Close!
$\{\boldsymbol{x}_2, \boldsymbol{x}_3, C = 1\}$
τ

(a) (b) (c) (d)

Figure 5.3: An example of the run time learning pipeline. The risk map $\rho(\boldsymbol{x})$ is updated as the path planning policy Π interacts the environment.

policy iteration process that underlies almost all RL algorithms [80]. The proposed approach utilizes this concept in order to enable run time occlusion aware navigation: as the robot gains experience about its current motion planning policy Π_k, it uses this experience to train quantiles $\{\theta_i(\boldsymbol{x})\}$ and find the risk map $\rho(\boldsymbol{x})$, which the motion planning policy Π_{k+1} uses to avoid high-risk states on the map.

Fig. 5.3 shows an example of this run time learning pipeline, in which the risk map $\rho(\boldsymbol{x})$, represented as the grid of cells, is updated at run time. In Fig. 5.3(a), the robot initially has no prior experience with this particular occlusion, and so $\rho(\boldsymbol{x}) = 0$ for these states (denoted by the color white). From (5.3), this means that the initial path generated is unaware of the risks associated with this occlusion, and simply cuts close to the corner. Fig. 5.3(b) shows the robot just before it rounds the corner, with an occluded dynamic obstacle approaching from the opposite direction. Fig. 5.3(c) shows the moment when it sees the dynamic obstacle for the first time within r_{vis}, meaning $C = 1$ according to (5.5). The data tuples $\{\mathcal{T}_i\}$ up to now are trained with QTD to find the q-quantiles $\theta_i(\boldsymbol{x})$, and updating the risk map $\rho(\boldsymbol{x})$ defined by (5.11). Fig. 5.3(d) shows the next time that the robot encounters this occlusion, along with the updated risk map. The path planning policy Π_{k+1} uses this updated $\rho(\boldsymbol{x})$ to minimize (5.3), now with $\rho(\boldsymbol{x}) > 0$ for the risky states, thereby planning a path that minimizes this risk.

Stability Properties

The stability properties of DRL algorithms are inherently difficult to characterize, due to the complexity of optimizing a cost function while simultaneously approximating the distribution of return costs [8]. Nevertheless, stability of the proposed approach can be established through the path cost (5.3) minimized by path planner Π. For a given starting state $\boldsymbol{x}_0$ and goal set $\mathcal{X}_g$, define T as the set of all feasible paths. Additionally, let ϱ represent the space of all feasible risk maps that satisfy (5.12). The path cost $S(\cdot)$ is a functional that maps the path τ and risk function $\rho(\cdot)$ to a positive scalar value. The optimal cost for this risk mapping S^*_ρ is defined as

$$S^*_\rho = \min_{\tau \in T} S(\tau, \rho(\cdot)) \tag{5.13}$$

Because $\rho \in [0, 1]$ as remarked in (5.12), we can likewise establish bounds for the value of S^*.

Lemma 5.1 *Define D_τ as the total length of feasible trajectory τ from initial state $\boldsymbol{x}_0$ to goal set $\mathcal{X}_g$. Furthermore, let τ_0 represent the trajectory found when $\rho(\boldsymbol{x}) = 0\ \forall \boldsymbol{x}$, i.e., occlusion-unaware motion. For a given risk map $\rho \in \varrho$, the optimal cost of (5.3), denoted as S^*_ρ, is bounded $S^*_\rho \in [S_{min}, S_{max}]$, where the lower bound $S_{min} = S_0$, and the upper bound $S_{max} = S_0 + \lambda D_{\tau_0}$.*

Proof: To show the lower bound, suppose $\rho = 0$ everywhere, so that $\int_\tau (\rho) d\tau = 0$ for all $\tau \in T$. This is equivalent to risk-agnostic motion, and Π will create a path τ_0 with optimal cost $S^*_\rho = S_0$. If $\rho > 0$ for some states, then $\int_\tau (\rho) d\tau \geq 0$ and $S^*_\rho \geq S_0$. Therefore, it must be that the minimum possible cost for a risk map $\rho \in \varrho$ is $S_{\min} = S_0$. To establish the upper bound, suppose $\rho = 1$ everywhere, so that $\int_\tau \rho d\tau = D_\tau$. In this case, the cost associated with τ_0 is $S = S_0 + \lambda D_{\tau_0}$. This means the optimal trajectory τ^* must have at most this cost, since any trajectory $\tau \neq \tau_0$ will have $D_\tau \geq D_{\tau_0}$ so that it cannot be the minimum cost. Additionally, if $\rho \leq 1$ for some states, it must be that $\int_\tau (\rho) d\tau \leq D_\tau$. Therefore, for a given $\rho \in \varrho$, the optimal cost is upper bounded $S^*_\rho \leq S_{\max} = S_0 + \lambda D_{\tau_0}$. ■

Further analysis requires additional information on how the risk map ρ changes over time. While this is difficult to characterize in a general way, it becomes more tenable if we assume, for a given state input $\boldsymbol{x}$, the risk map $\rho(\cdot)$ is non-decreasing, a reasonable assumption since the upper-end quantile values are designed to be disproportionately affected by negative outcomes ($C = 1$) more than positive outcomes ($C = 0$). Concretely, let $\rho(\cdot)$ be the risk map used by Π to generate risk-aware motion. The proposed approach uses $\rho(\cdot)$ to navigate through the environment, collecting data to help train an updated risk map $\rho_+(\cdot)$, so in general $\rho_+(\cdot) \neq \rho(\cdot)$. We define this non-decreasing property as:

$$\delta\rho = \rho_+(\boldsymbol{x}) - \rho(\boldsymbol{x}) \geq 0, \quad \forall \boldsymbol{x} \in \mathcal{X}. \tag{5.14}$$

Intuitively, this describes a situation in which the estimated risk function ρ only increases the estimated risk as more data is collected. If the approach initially assumes no risk, then $\rho = 0$ to start and over time will naturally only increase $\rho(\cdot)$ as negative outcomes are experienced. Or, it may be that a conservative estimate of ρ is desired and (5.14) is enforced during the QTD training. In either case, if this assumption is satisfied, then we can make the following claim:

Lemma 5.2 *For a given starting state $\boldsymbol{x}_0$ and goal set $\mathcal{X}_g$, define S^*_ρ as the optimal path cost associated with risk map $\rho(\cdot)$. If $\rho_+(\cdot)$ defines the updated risk map based on new information and* (5.14) *is satisfied, then the updated optimal cost $S^*_{\rho_+}$ is also non-decreasing so that $S^*_{\rho_+} \geq S^*_\rho$.*

Proof: For a given trajectory $\tau \in T$, it can be shown that difference in cost $\delta S_\tau = \lambda \int_\tau (\delta\rho) d\tau$. Since $\delta\rho \geq 0$, this means that $\delta S_\tau \geq 0$ for all feasible trajectories. Define the optimal trajectories associated with optimal costs S^*_ρ and $S^*_{\rho_+}$ as τ^* and τ^*_+, respectively. If $\tau^* = \tau^*_+$, then it has been shown that $\delta S_{\tau^*} \geq 0$. However, if $\tau^* \neq \tau^*_+$, then it must be that $S^*_{\rho_+} \geq S^*_\rho$. This is shown by contradiction: if $S^*_{\rho_+} < S^*_\rho$ and $\delta S_{\tau^*_+} \geq 0$, then $S(\tau^*_+, \rho) < S^*_\rho$. This is not possible, however, since by definition $S^*_\rho = S(\tau^*, \rho) \leq S(\tau^*_+, \rho)$. Therefore, it must be that $S^*_{\rho_+} \geq S^*_\rho$.

■

Since the cost S is both lower- and upper-bounded via Lemma 5.1 and, under the non-decreasing assumption defined in (5.14), the cost can only increase via Lemma 5.2, then the cost must eventually converge to a finite value. Although multiple trajectories could theoretically have this same cost in certain situations, it is usually not a problem in practice for a path planner to find and stick with a certain trajectory, implying the planned path itself will effectively converge as well.

5.4 Simulations

Simulations were performed to validate the ability of the proposed approach to adapt its planned path at run time and promote visibility of dynamic obstacles through occlusion-aware motion. Gazebo was used as a high fidelity simulator for the Clearpath Jackal platform, with a state $\boldsymbol{x} = [p_x, p_y, \theta]^T$ defined by global Cartesian coordinates (p_x, p_y) and heading θ, and controls $\boldsymbol{u} = [v, \omega]^T$. Different environments were constructed in Gazebo, and the robot was equipped with a 360° lidar to produce laser scans of the immediate surroundings. The SLAM package Gmapping [67] was used to created a map of static portions of the environment, while AMCL [3] was used to localize within these maps. A ROS obstacle detection package [64] observed the dynamic obstacles from the current laser scan, encoded as a set of circles $\mathcal{O} = \{(\boldsymbol{o}_i, r_{o,i})\}$, each defined by a center $\boldsymbol{o}$ and radius r_o. To simulate realistic scenarios, the dynamic obstacles moved along pre-determined paths throughout the environment, controlling their acceleration to either track a desired speed or slow down when the robot was observed to be along their path. Velocity information about the dynamic obstacles was not inferred from their location, both for simplicity in analysis and in order to highlight how the proposed approach is agnostic to an intention-aware mechanism for the dynamic obstacles.

The navigation stack used by the robot is shown in Fig. 5.1 and is composed of two modules: (i) a high-level global path planner that produces a feasible trajectory $\boldsymbol{\tau}$ for the robot to follow towards a goal, and (ii) a low-level controller that produces controls $\boldsymbol{u}$ to track this trajectory. A Hybrid A* planner [26] was used for the global planner and ran at a frequency of 0.5 Hz, while a Dynamic Window Approach (DWA) controller [33] was used as the low-level controller and ran at 50 Hz. The DWA controller commanded $v \in [-0.2, 0.8]$ m/s and $\omega \in [-2.0, 2.0]$ rad/s. Both modules had access to an occupancy map and dynamic obstacle encoding $\mathcal{O}$ in order to perform typical collision avoidance. Both the ego robot and the dynamic obstacles could only observe each other when within line-of-sight, i.e., unoccluded by static obstacles in the environment.

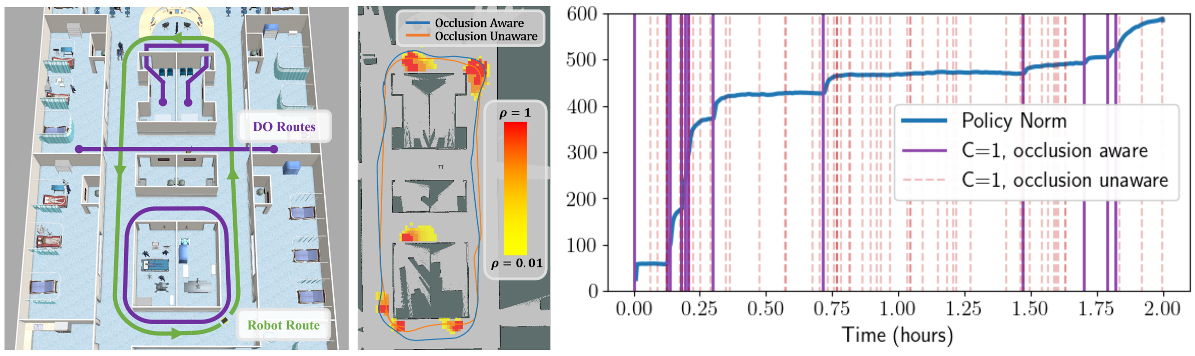
Our approach augments this path planner through the use of a risk map $\rho : \mathbb{R}^3 \to \mathbb{R}$ that is learned at run time to produce occlusion-aware navigation. This risk map utilizes a discretized approximation $\mathcal{X}_\text{d}$ of the true global state space $\mathcal{X}$, where each $\boldsymbol{x}_\text{d} \in \mathcal{X}_d$ represents a small grid cell of the state space, centered on $\boldsymbol{x}_\text{d}$ and sized 0.25 m in the the xy cartesian position, and $\pi/4$ rad in heading. To start, $\rho(\cdot) = 0$ for all bins, resulting in occlusion-unaware motion initially. Data tuples

$\mathcal{T}_i = \{\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, C_i\}$ were collected into a replay buffer $\mathcal{R}$ with a maximum size of 1e5 at a rate of 2 Hz, so that it would take over 13.5 hours to completely fill the buffer. Data from this replay buffer were used to learn a return-cost distribution $\hat{\eta}(\boldsymbol{x})$ defined by $m = 32$ q-quantile parameters $\{\theta_i(\boldsymbol{x})\}$ for each binned global pose. On a Lenovo Thinkpad X1 with an i7 12-core processor, the QTD algorithm that calculated $\{\theta_i(\boldsymbol{x})\}$ could be iterated 10,000 times in 34 ms, enabling new data to be incorporated quickly into the risk map over environments of reasonable size. The risk map $\rho(\cdot)$ was then created by constructing cost with the the mth quantile of each bin, which tracked the worst 2% of return costs for each global pose (p_x, p_y, θ).

Discussed below are two scenarios that serve to highlight the benefits of the proposed approach. The first is a hospital environment in which the robot must perform a patrol-like pattern throughout the main floor that is populated with dynamic obstacles moving from room to room. The second is a warehouse that contains two dynamic obstacles that moves much faster than the robot and must slow down when the robot is crossing over its forward path. Together, the results of these scenarios showcase how occlusion-aware navigation can improve overall navigation of both the ego robot as well as the dynamic obstacles.

5.4.1 Hospital Environment

Fig. 5.4(a) shows a picture of the Gazebo environment, along with the routes taken by the robot and the dynamic obstacles throughout the environment. The many walls and rooms of the hospital served to occlude the presence of dynamic obstacles from the robot, which resulted in instances where the robot suddenly observed a dynamic obstacle close by, requiring the navigation stack to quickly replan motion around these obstacles. Twelve different dynamic obstacles moved throughout



(a) (b) (c)

Figure 5.4: Hospital simulation environment for validating the proposed occlusion-aware navigation. Fig. 5.4(a) shows the routes of the robot and the dynamic obstacles. Fig. 5.4(b) visualizes the learned risk map after two hours of simulation, and Fig. 5.4(c) shows how this policy norm changes over time.

the environment, each with a radius of 0.5 m and speed within the range $0.3 - 0.8$ m/s. It was empirically found that a distance of 2 m was ample space for the robot navigation stack to observe and smoothly replan around dynamic obstacles, so $r_{\text{vis}} = 2$ in (5.5). With our proposed approach, a risk map $\rho(\cdot)$ was learned from direct experience over the course of a two-hour simulation. Fig. 5.4(b) visualizes this risk map by showing $\rho_{\text{2D}}(p_x, p_y) = \max_\theta \rho(p_x, p_y, \theta)$ for each binned global position (p_x, p_y). It can be seen how areas of high risk ($\rho = 1$) are learned around the occluding corners of the map, where the distance between the robot and a newly observed obstacle is below $r_{\text{vis}} = 2$. To get a sense of how this risk map changes over time, define the policy norm $|\rho(\cdot)|$ as the sum of risk values over all bins,

$$|\rho(\cdot)| = \sum_{\boldsymbol{x}_\text{d} \in \mathcal{X}_\text{d}} \rho(\boldsymbol{x}) \tag{5.15}$$

Fig. 5.4(c) shows how the norm of $\rho(\cdot)$ evolves over the course of the two hours. Vertical purple lines show time instances in which $C = 1$, which precede an increase in $|\rho(\cdot)|$, showing how the QTD algorithm quickly incorporated this new information into the risk map. Over time, the frequency of $C = 1$ instances decreases as the global planner uses the updated $\rho(\cdot)$ to avoid the risky states. This can be compared to the faint red lines, showing the results of a navigation stack that did not use $\rho(\cdot)$ to inform its policy, instead remaining occlusion-unaware and repeatedly encountering $C = 1$.

Fig. 5.5(a) shows a normalized histogram of the command robot speed v over the entire two hours, both with the occlusion-aware run time learning and without such a learning enable component. It can be seen how the occlusion-aware navigation stack resulted in higher overall frequency of the highest allowable commanded speed of 0.8 m/s. This a direct result of the improved visibility of dynamic obstacles around occlusions, as this prevented instances where the robot suddenly sees a dynamic obstacle close by, and must rapidly stop (or even reverse) in order to negotiate around these obstacles with the desired padded radius. Additionally, Fig. 5.5(b) shows a normalized histogram of the commanded robot angular velocity ω, which tells a similar story: with occlusion-aware navigation, the robot moves along a straight path ($\omega \approx 0$) with a higher frequency when compared to occlusion-unaware navigation since it does not need to rapidly move out of the way of newly-observed dynamic obstacles.

5.4.2 Warehouse Environment

Fig. 5.6(a) shows a picture of the warehouse in which two dynamic obstacles moved between each of the inner rooms in a clockwise direction. The first dynamic obstacle (DO1) moved with a maximum speed of 4 m/s, and a second dynamic obstacle (DO2) moved with a maximum speed of 6 m/s. The route of the robot moved around these rooms and crossed over the route of the dynamic obstacles in several locations. Occlusion-aware navigation was learned through data collected at

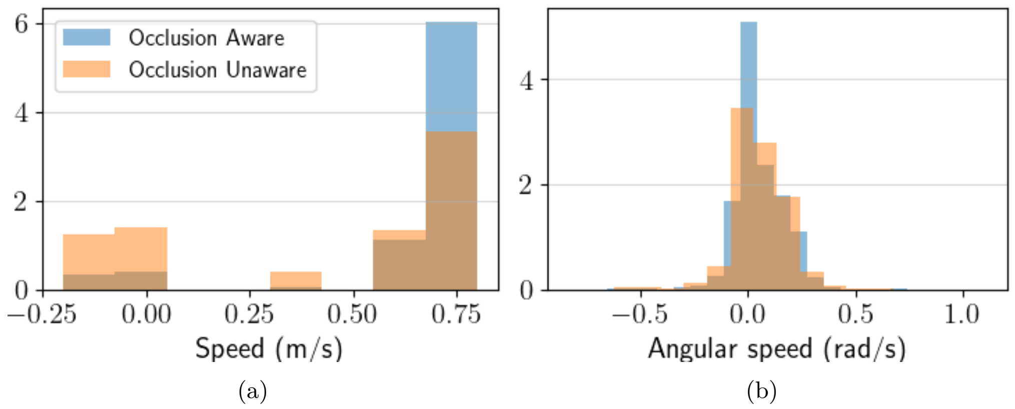


Figure 5.5: Normalized histograms of the commanded speed v and angular velocity ω over the two hour simulation in the hospital environment of Fig. 5.4.

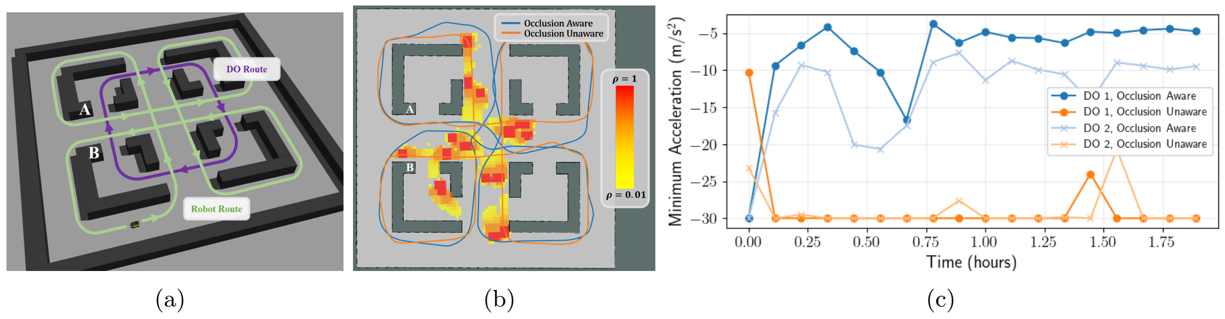


Figure 5.6: Warehouse simulation environment for validating the proposed occlusion-aware navigation. Fig. 5.6(a) shows the routes of the robot and the dynamic obstacles. Fig. 5.6(b) visualizes the learned risk map after two hours of simulation, and Fig. 5.6(c) shows minimum decelerations of the two dynamic obstacles over time.

runtime, producing the ρ_{2D} shown in Fig. 5.6(b). Also shown are the route of the robot around the warehouse, both with occlusion-aware navigation (blue line) and without occlusion aware navigation (orange line). Fig. 5.6(a) and 5.6(b) both label doorway A into which dynamic obstacles enter and doorway B from which dynamic obstacles emerge. The occlusion-aware policy actually adapted to this asymmetry by learning to move away from B and closer to A in order to improve the visibility of dynamic obstacles, while the occlusion-unaware policy remained close to B in order to reduce total path length. The same occlusion-aware behavior also developed near other doorways in the environment as well.

In this scenario, the occlusion-aware path serves to improve the overall navigation of DO1 and DO2. As mentioned previously, when the dynamic obstacle observed the ego robot along its forward route some distance d ahead, a negative acceleration a (i.e., deceleration) was applied that allowed the dynamic obstacle to stop before collision with the robot. Kinematically, a body in motion travelling at speed v must apply the following acceleration in order to fully stop:

$$a = v^2/2d. \tag{5.16}$$

The above equation shows how a higher deceleration is required to slow down when a dynamic obstacle first observes the robot to be a short distance away d in front of it. Since DO1 and DO2 are moving at a relatively high speed compared to the ego robot, the majority of the interactions consist of the dynamic obstacles needing to slow down and wait for the ego robot to move out of their way before they can travel full speed again. This means that occlusion-aware navigation of the ego robot can *also benefit the dynamic obstacles by observing the ego robot at a farther distance away.* Since both DO1 and DO2 observe the robot at a farther distance d, this requires a lower magnitude of deceleration in order to avoid collisions with the robot.

Fig. 5.6(c) showcases this principle by plotting the deceleration a of both dynamic obstacles when the robot followed an occlusion-aware policy, as well as an occlusion-unaware policy. In order to improve readability, the minimum deceleration over a sliding window of 6 minutes is shown and was capped at -30 m/s^2. It can be seen that over the course of the simulation, both dynamics obstacles were required to brake harder if the robot followed an occlusion-unaware policy than if the robot had learned an occlusion-aware policy. Additionally, DO1 experiences a smaller magnitude deceleration than DO2 under the occlusion-aware policy, since DO1 traveled at a smaller speed.

5.5 Experiments

Experiments were conducted on a Boston Dynamics Spot [12] quadruped robot to further validate the proposed approach in a real-world setting. Spot was equipped with an OS1 Ouster 3D lidar sensor, using Gmapping to map the static obstacles of the real-world environment a priori, AMCL to localize within this environment, and the same dynamic obstacle detection pipeline as the simulations. Additionally, the same navigation stack from the simulated experiments was used with Spot, using a Hybrid A* planner to create a global path to follow and a DWA controller to track this trajectory. As Spot navigated the environment, dynamic obstacle observations were used to determine outcome C_i and create data tuples $\mathcal{T}_i = \{\boldsymbol{x}_{i-1}, \boldsymbol{x}_i, C_i\}$ which were collected into a replay buffer and used to train a risk map $\rho : \mathbb{R}^3 \to \mathbb{R}$ at run time. This risk map informed the Hybrid A* planner of areas to avoid, thereby creating an occlusion-aware navigation policy. Lastly, Spot was outfitted with a Spot Core [78] computational payload, allowing all modules within the proposed navigation stack to run onboard Spot.

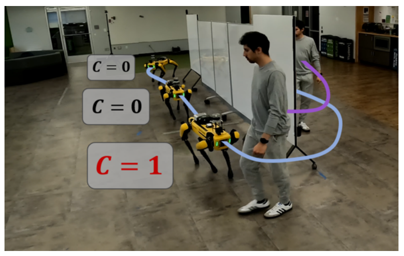
Two experiments were conducted, each highlighting different aspects of the proposed approach. The first placed Spot within a small-scale map that had a single occluding wall, focusing on how the proposed approach creates an occlusion-aware policy by learning from negative outcomes, even if they are infrequent. The second tests the navigation stack in a real-world office space, showcasing the benefits of the proposed approach on a real-world system.

5.5.1 Small-Scale Map

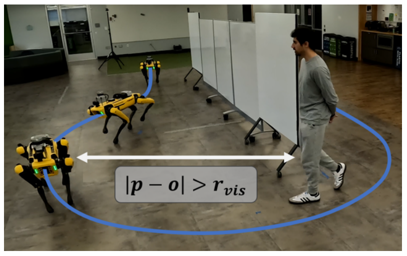
The robot was placed in a small-scale map with a single wall that acted as an occluding median, around which the robot was tasked to move in a counterclockwise direction. Fig. 5.7(a) shows a picture of this environment, as well as snapshots of the robot moving with initially occlusion-unaware motion. After four full laps around the median, a human actor was introduced as a dynamic obstacle within this map. This actor was asked to adopt an adversarial policy of appearing from behind the occlusion as close as possible to the robot (also shown in Fig. 5.7(a)), testing the worst-case scenario in this experiment. This triggered a negative outcome of $C = 1$, information which the proposed approach incorporated, resulting in a trajectory that improved visibiliy around the occluding corner, shown in Fig. 5.7(b). Additionally, Fig. 5.7(c) visualizes this risk map after several instances of negative outcomes, as well as a comparison of trajectories before and after experiencing these negative outcomes.

5.5.2 Office Space

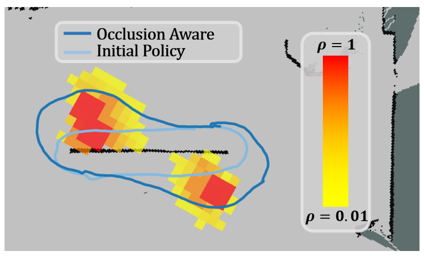
Additional experiments were conducted in a real-world office space setting in which Spot was tasked to navigate. Again, a human actor was instructed to move against Spot's direction of travel and to appear as close to Spot as possible around two different corners within the environment. The goal was to test the worst-case scenario and see how the proposed approach responded. Fig. 5.8(a) shows



(a)

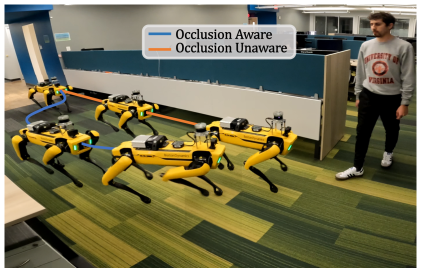


(b)

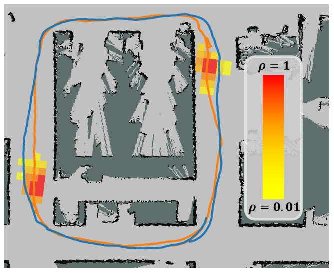


(c)

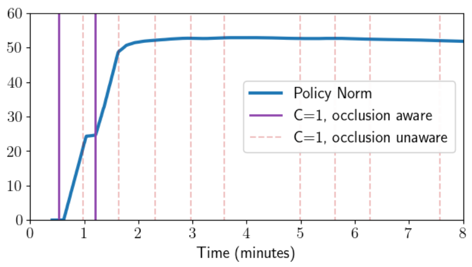
Figure 5.7: Small-scale environment used to validate the run time learning capabilities of the proposed approach. Fig. 5.7(a) shows a portion of this environment, along with snapshots of the initial occlusion-unaware path and an instance when a dynamic obstacle was observed for the first time too close to the robot. Fig. 5.7(b) shows the final learned path within the environment. Fig. 5.7(c) shows the resulting risk map as well as a comparison of how the path adjusts to the learned risk.



(a)



(b)



(c)

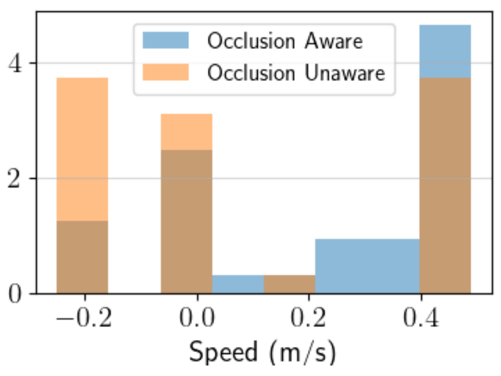
Figure 5.8: Office space environment used to validate the run time learning capabilities of the proposed approach. Fig. 5.8(a) shows a portion of this environment, along with snapshots of the initial occlusion-unaware path. Fig. 5.8(b) visualizes the learned risk map, as well as the resulting occlusion-aware path. Fig. 5.8(c) shows how the policy norm changes over time.

the camera view of one of these corners, as well as occlusion-unaware motion and occlusion-aware motion (as learned by the proposed approach). As can be seen within this figure, the proposed approach allows Spot to adjust its position and gain a better visibility around the occluding corner. Fig. 5.8(b) shows the risk map learned by the proposed approach, as well as the resulting trajectories. Fig. 5.8(c) visualizes the policy norm (blue line) over the duration of the experiment, showing how with two negative outcomes (purple lines) the run time learning stack changed the risk map to reflect these experiences and eliminate any more negative outcomes. This is also compared against the same exact experiment with the learning stack disabled so that Spot's motion remains occlusion-unaware during the experiment. The faint red lines show instances of $C = 1$ in this case, which continue unmitigated from the occlusion-unaware path Spot continues to take.

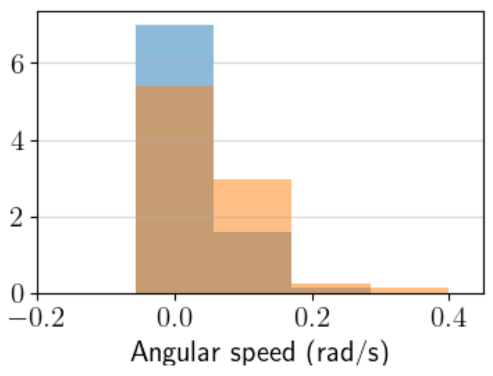
Additionally, Fig. 5.9 shows a normalized histogram of the controls $\boldsymbol{u} = [v, \omega]$ commanded by the DWA controller over the course of the experiment. As was the case in the simulation experiment described in Sec. 5.4.1, the data show two distinct trends: first, the occlusion-aware motion policy allows the DWA to commanded higher speeds more frequently, owing to the fact it improves visibility of dynamic obstacles and reduces the need to slow down or stop because they are observed at an appropriate distance away. Second, the occlusion-aware motion policy allows a higher frequency of $\omega \approx 0$, meaning Spot must turn less frequently as well, again owing to the improved visibility of dynamic obstacles around occlusions.

5.6 Discussion and Conclusion

In this chapter we have outlined an approach for occlusion-aware navigation which learns risk-aware policies from direct experience with the environment. This allows for an occlusion-aware policy that is bespoke to the idiosyncrasies of real-world environments that are difficult to model a priori,



(a)



(b)

Figure 5.9: Normalized histograms of the commanded speed v and angular velocity ω over the course of the office space experiment in Fig. 5.8.

such as dynamic obstacle size, traffic flow, and motion policy. The approach is designed to be risk-sensitive and data-efficient, enabling an occlusion-aware policy that is learned from relatively few negative outcomes. The result is a policy that learns from its previous experiences and moves to improve visibility around occlusions which have shown to be problematic in the past.

Despite this flexibility, there are particular challenges with using the QTD algorithm to learn quantiles from experience at run time. For example, any RL technique assumes the Markov property in which the distribution of cost variable C is entirely dependent upon the current robot state $\boldsymbol{x}$ and the current motion policy. Effectively, this means that the chance of encountering a negative outcome $C = 1$ must be entirely dependent upon the current global pose of the robot. While this may be true for simple environments, modeling actual human motion may not be as straightforward, as the speed and traffic patterns of dynamic obstacles may be influenced by either the historic motion of the robot or even complex external factors not captured by the robot state $\boldsymbol{x}$, e.g., traffic patterns that change based on amount of traffic flow or time of day. Additionally, this work assumes the underlying dynamic obstacle probabilities are constant over time for a given environment, which may not be true for a real-world deployment. If these probabilities do change over time, then the current approach will conflate data collected from these different periods of time, producing quantiles $\rho(\cdot)$ that do not accurately describe the actual risk. This is related to the question of how much experiential data to keep as the robot is deployed for extended periods of time. If the traffic patterns change from hour-to-hour or day-to-day, it may be disadvantageous to keep all data within a single replay buffer. Instead, it may be better to purge the replay buffer and restart the QTD algorithm with freshly collected data, or the replay buffer may only be partially purged, with certain datum recognized as important and kept. Identifying these transition periods is a particular challenge for this approach, and is left in this work as an interesting avenue for future research.

Although this framework approaches a stable risk-aware policy over time, it relies upon the robot experiencing a negative outcome before adjusting the risk metric. Before moving near an occlusion, however, the robot may observe dynamic obstacles emerging from this occluded region. This observation suggests one potential approach may leverage these dynamic obstacle observations and infer the risk of a negative outcome without the need to experience it. This idea is the basis for Chapter 6, which extends the idea of risk-based, data-informed occlusion-aware navigation presented in this chapter towards a policy that can infer the risk of negative outcomes from historical observations of dynamic obstacles.

Chapter 6

What's the Worst That Can Happen? Run Time Data-driven Occlusion-Aware Navigation

6.1 Introduction

Chapter 5 introduced an approach to occlusion-aware navigation that cast the problem as risk-aware navigation, where this risk was caused by the uncertainty created by occluded dynamic obstacles. This risk was generated by negative outcomes directly experienced by the robot in particular locations throughout the global map, and eventually the proposed framework learned to avoid such areas, effectively resulting in an occlusion-aware policy. Despite the effectiveness of this policy over time, it should be noted that when the robot is far away from an occluding corner, the robot may observe dynamic obstacles emerging from behind this corner *before* the robot experiences these negative outcomes. An intelligent policy may take these observations and understand the possibility of negative outcomes given the historical record of dynamic obstacles emerging from this occluded region, combined with an understanding of what would happen if the robot traveled close to the occlusion. This chapter explores an occlusion aware policy that can understand the "what if" scenarios given previous dynamic obstacle observations within the environment. In this way, an occlusion-aware policy may be developed without the need for the robot to experience the negative outcomes associated with occlusions.

Concretely, this chapter provides several novel contributions towards run time occlusion-aware navigation. First, the risk-based occlusion-aware concepts developed in Chapter 5 are further extended, with a focus on how unknown occluded obstacles present a risk of increased travel time. Once these concepts are formalized, a framework is developed that can calculate this risk for a given path. This risk calculation is based upon two different models: (i) an offline-trained supervised learning model that predicts how well the robot may negotiate a recently unoccluded dynamic obstacle, and (ii) an online-created model that predicts the probability of encountering such a situation given the historical observations of dynamic obstacles within the environment.

These components are combined to create a risk-sensitive path-planning cost function used by a graph-search algorithm to find a path that minimizes this cost. When constructing the risk-sensitive cost function, the underlying cost distribution is assumed to take form of a parametric function. This is done so that these distributions may be simple to define with parameter values inferred from data collected at run time, as well as ensure the cost function is computationally quick to evaluate within the graph-search algorithm.

6.2 Problem Formulation

Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ represent the equations of motion for a robot with state $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and control inputs $\boldsymbol{u} \in \mathbb{R}^{n_u}$. These controls are produced by a low-level controller that is tracking a position-based path $\boldsymbol{\tau} \in \mathbb{R}^2$ generated at time t_0. Additionally, define $\mathcal{E}$ the set of all obstacles within the environment to be avoided. In this chapter , the goal is to reduce the estimated travel time $T_{\boldsymbol{\tau}}$ needed to track $\boldsymbol{\tau}$ that provides a path for the robot towards a goal $\boldsymbol{g} \in \mathbb{R}^2$, while avoiding obstacles within $\mathcal{E}$. The types of obstacles encountered in $\mathcal{E}$ can be decomposed into two distinct categories:

- Static map $\mathcal{M}$ that represents all stationary obstacles, e.g. walls, and may be mapped offline.
- The set of dynamic obstacles $\mathcal{D}$, that must be observed at run time.

Due to the line-of-sight nature of the sensors available to the robot, only a subset of dynamic obstacles $\mathcal{D}_o \in \mathcal{D}$ are actually observable at any given moment, meaning the set of unobserved obstacles $\mathcal{D}_u = \mathcal{D} \setminus \mathcal{D}_o$ is unknown at run time. Both $\mathcal{M}$ and $\mathcal{D}_o$ may be used to inform the creation of a feasible path $\boldsymbol{\tau}$, which takes some finite time $T_{\boldsymbol{\tau}}$ to track. However, the uncertainty introduced by $\mathcal{D}_u$ can be difficult to reconcile, as the position, velocity and amount of unobserved obstacles can have a huge impact on $T_{\boldsymbol{\tau}}$. From the point of view of the robot, this epistemic uncertainty ultimately causes $T_{\boldsymbol{\tau}}$ to be non-deterministic, as the ego robot may need extra time to avoid obstacles that were previously occluded. Thus, we may consider $T_{\boldsymbol{\tau}}$ to effectively be a *stochastic* variable, and assume it has some underlying distribution $P(T_{\boldsymbol{\tau}})$. Furthermore, we assume that $\mathcal{D}_u$ is the only factor that contributes to the distribution of $T_{\boldsymbol{\tau}}$ and not other potential factors, e.g., unmodeled robot dynamics. This leads to the first problem addressed in this paper:

Problem 1: ***Producing a Risk-Sensitive, Time-Optimal Path:*** For a given static obstacle map $\mathcal{M}$ and observed dynamic obstacle set $\mathcal{D}_o$, let $P(T_{\boldsymbol{\tau}}) \in \mathcal{P}$ represent the distribution of possible times to track $\boldsymbol{\tau}$, caused by the underlying epistemic uncertainty of $\mathcal{D}_u$. Furthermore, define a risk metric $\rho : \mathcal{P} \to \mathbb{R}$ that maps from this distribution to a real number. We seek a policy Π that creates a path $\boldsymbol{\tau}$ to minimize this risk metric:

$$\boldsymbol{\tau} = \arg\min_{\boldsymbol{\tau}'} \rho(\boldsymbol{\tau}') \tag{6.1}$$

The distribution $P(T_{\boldsymbol{\tau}})$ must be well characterized in order to effectively perform the minimization in (6.1). For example, if the robot knows there are no dynamic obstacles in the environment, then $T_{\boldsymbol{\tau}}$ is effectively deterministic, and solving (6.1) amounts to the standard path-planning problem in a known environment $\mathcal{M}$. However, if the environment contains many dynamic obstacles, e.g., a busy airport, then $P(T_{\boldsymbol{\tau}})$ could be very much affected by the location or time of day. This presents another reasonable requirement that the proposed approach should adapt at run time to different traffic patterns inferred from previous dynamic obstacle observations.

Problem 2: ***Adaptability using Run Time Observations:*** Define $\{O_i\}$ as a series of run time dynamic obstacle observations, where each O_i represents dynamic obstacle information gathered by the robot within the environment in which it is deployed. We seek a method that can inform the stochastic distribution $P(T_{\boldsymbol{\tau}})$ at run time using these observations,

$$\{O_i\} \rightarrow P(T_{\boldsymbol{\tau}}), \tag{6.2}$$

allowing the approach a high degree of adaptability to different $\mathcal{D}$ that may be present in a given environment.

6.3 Approach

Fig. 6.1 shows a diagram of the proposed approach, in which run time observations $\{O_i\}$ are used to create a path $\boldsymbol{\tau}$ from current position $\boldsymbol{p}_0$ to goal position $\boldsymbol{g}$. To create a risk-sensitive policy for run time occlusion-aware motion planning, the proposed process must (i) model the distribution $P(T_{\boldsymbol{\tau}})$ given $\{O_i\}$, and (ii) use this model within an optimal path planning algorithm. In order to model $P(T_{\boldsymbol{\tau}})$, the proposed approach separates the process into two distinct sub-models: a model created online, and a model created offline. The online-created model infers the distribution $P(O, n)$ that describes the probability of sensing n dynamic obstacles at a given location and speed. This is not only dependent on where the robot is located, but also the size, speed and traffic pattern of the dynamic obstacles within the environment. Observations $\{O_i\}$ can thus inform $P(O, n)$, contributing to a sense of where dynamic obstacles may be observed in the future given where they have been observed historically. This is used in conjunction with a second model, $P(T_{\boldsymbol{\tau}}|O, n)$, that describes how such dynamic obstacle observations may effect travel time $T_{\boldsymbol{\tau}}$ for a given trajectory. In particular, focus is given to dynamic obstacles that emerge from occluded regions. This model is created offline by generating simulations of interactions between robot and previously occluded dynamic obstacles. Both the online and offline models are used together within the path planner to solve (6.1), producing a trajectory $\boldsymbol{\tau}$ that is tracked by a low level controller.

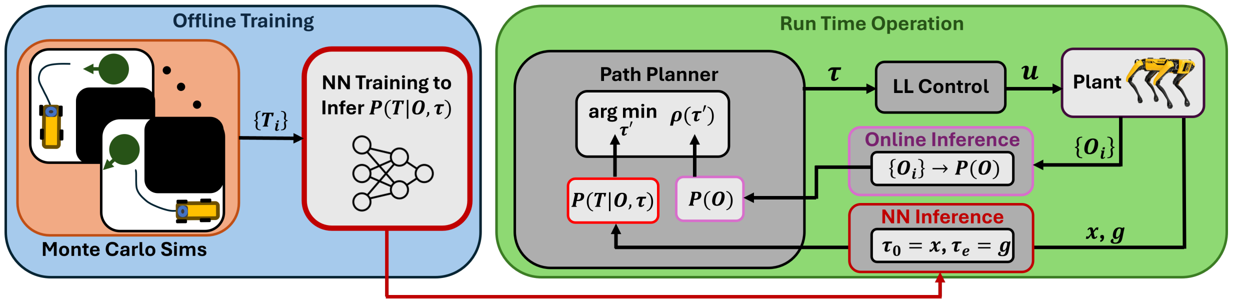


Figure 6.1: Block diagram for both the offline training stage and online solver failure prediction and recovery framework.

Sec. 6.3.1 formalizes the notion of $P(T_{\boldsymbol{\tau}})$, as well as how this distribution may be decomposed into two different factors that are separately modeled. Sec. 6.3.2 describes how observations $\{O_i\}$ are used to create a model of $P(O, n)$ at run time, while Sec. 6.3.3 details how a model of $P(T_{\boldsymbol{\tau}}|O, n)$ may be trained offline. Lastly, Sec. 6.3.4 describes how both models may be used together to solve (6.1) at run time via a graph-search path planning algorithm.

6.3.1 Formalizing Distribution of Tracking Time

In order to facilitate discussion, let us first define $\boldsymbol{\tau}$ as a series of $\{x, y\}$ locations $\boldsymbol{p}_i$ for the robot to track throughout the environment, so that $\boldsymbol{\tau} = \{\boldsymbol{p}_i\}$. The total time to traverse the entire path $\boldsymbol{\tau}$ is equal to the sum of times required to traverse each segment of this path,

$$T_{\boldsymbol{\tau}} = \sum_i T_i, \tag{6.3}$$

where T_i is the time required for the robot to move from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$. Each T_i can be further broken down into a deterministic and stochastic component:

$$T_i = T_i^0 + \Delta T_i \tag{6.4}$$

Here, T_i^0 is the time required to move from from $\boldsymbol{p}$ to $\boldsymbol{p}_{i+1}$, assuming no dynamic obstacles are occluded in the environment. This is considered a deterministic variable, as it is influenced by factors such as robot kinematic/actuation constraints and known obstacles in the environment. On the other hand, ΔT_i is labeled the *excess time* and is a stochastic variable that contains the influence of the epistemic uncertainty caused by occlusions potentially hiding other dynamic obstacles $\mathcal{D}_u$ in the environment. In this way, $T_{\boldsymbol{\tau}}$ may be separated out as well:

$$T_{\boldsymbol{\tau}} = T_{\boldsymbol{\tau}}^0 + \sum_i \Delta T_i \tag{6.5}$$

Here, $T^0_{\boldsymbol{\tau}} = \sum T^0_i$ is the deterministic time to complete $\boldsymbol{\tau}$, and there are many standard techniques to find the minimum-time path through a known environment. The inclusion of $\sum_i \Delta T_i$ complicates this process because the distribution of ΔT_i is to be inferred at run time from dynamic obstacle observations $\{O_i\}$, and so any approach must be adaptable enough to account for information gathered at run time.

Furthermore, we assume that each dynamic obstacle observation O_i contains information on the position $\boldsymbol{p}_{Oi}$ and velocity $\boldsymbol{v}_{Oi}$ of that dynamic obstacle, which may be easily inferred from any typical line-of-sight sensor:

$$O_i = [\boldsymbol{p}_{Oi}, \boldsymbol{v}_{Oi}]^\intercal \tag{6.6}$$

Such an observation must necessarily occur within the FOV $\mathcal{F}$ of the robot, and our approach is interested in dynamic obstacles that are newly observed emerging from occluded areas along the boundary $\delta\mathcal{F}$ along unoccupied space, typically defined by large jumps in lidar range measurements between successive angles. To help formalize this important distinction, we make the following definition:

Definition 1: For some observation time window Δt_O, we characterize the observation of an obstacle as $\Omega \in \{0, 1\}$, where $\Omega = 1$ denotes an obstacle that has emerged from an occluded area and is newly observed, otherwise $\Omega = 0$.

Here, a time window Δt_O must be defined because Ω describes a discrete event that occurs over a continuous time. We can further define $O_\Omega = [O \wedge (\Omega = 1)]$ as observing a dynamic obstacle emerging from an occluded area at location and velocity O. It is also important to recognize that during Δt_O, multiple dynamic obstacles may emerge from the same location $\boldsymbol{p}_O$, depending on the length of the observation window. Although a rigorous treatment would consider each dynamic obstacle as its own stochastic variable, this would result in a stochastic number of stochastic variables, which is analytically difficult to handle for run time optimization techniques. Instead, we make the assumption that dynamic obstacles emerging from an occlusion at location $\boldsymbol{p}_O$ will realistically have similar velocity distributions. Thus, we define n_Ω as the number of obstacles that are newly observed from occluded regions.

With these concepts defined, we seek to describe $P(T_{\boldsymbol{\tau}})$, the distribution of possible travel times while tracking a total path $\boldsymbol{\tau}$, via individual distributions $P(T_i)$ defined over each segment. Furthermore, focus is placed on describing the stochastic portion of this travel time segment $P(\Delta T_i)$. In the proposed approach, the distribution $P(\Delta T_i)$ is cast as a marginal over the following factors:

$$P(\Delta T_i) = \sum_{O_\Omega, n_\Omega} P(\Delta T_i | O_\Omega, n_\Omega) P(O_\Omega, n_\Omega) \tag{6.7}$$

Here, $P(\Delta T_i | O_\Omega, n_\Omega)$ is a factor determines how the excess time depends on the dynamic obstacle

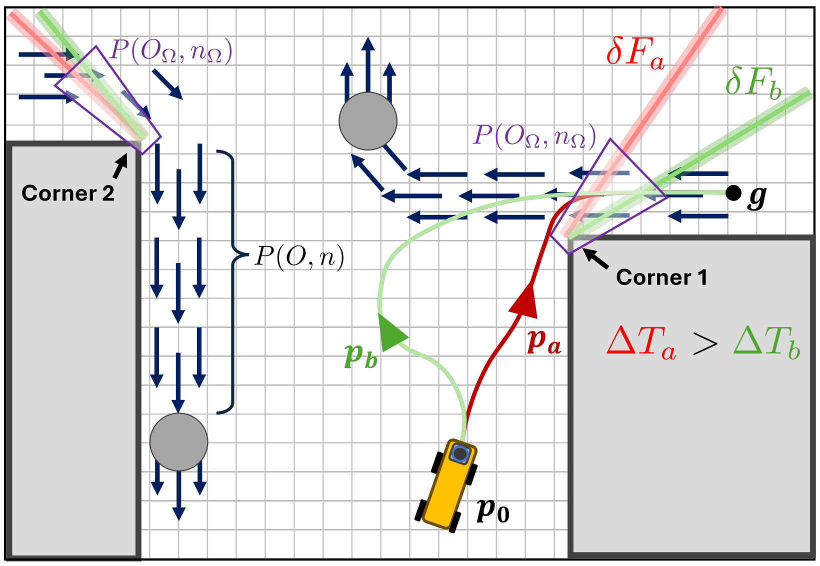


Figure 6.2: Motivating example of the factors that influence $P(T_{\boldsymbol{\tau}})$.

observation O_Ω as well as the number of dynamic obstacles n_Ω. The second factor $P(O_\Omega, n_\Omega)$ determines the probability of making this observation in the first place. Both factors $P(\Delta T_i|\cdot)$ and $P(O_\Omega, n_\Omega)$ also have a dependence on the locations $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ that define the path segment, but is not explicitly shown for notational brevity.

Fig. 6.2 shows an example how these two factors are important in planning a path from $\boldsymbol{p}_0$ to $\boldsymbol{g}$. Two example paths are shown, with locations $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ chosen along each, as well as the FOV edge $\delta\mathcal{F}$ that defines the occluded regions. The goal $\boldsymbol{g}$ is located behind occluding corner 1, and there is a non-zero probability of observing a dynamic obstacle emerging from around both occluding corner 1 and 2. This probability, $P(O_\Omega, n_\Omega)$, is modeled online from previous observations $\{O_i\}$ that help create a more general probability of observing a dynamic obstacle $P(O, n)$. In Fig. 6.2, both $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ have similar $P(O_\Omega, n_\Omega)$, since their δF FOV edges intersect this region of non-zero probability. If a dynamic obstacle does emerge from behind corner 1, $\boldsymbol{p}_a$ will incur a higher excess time than $\boldsymbol{p}_b$ since $\boldsymbol{p}_a$ is closer to this corner. Additionally, if a dynamic obstacle does emerge from corner 2, both $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ are far enough away as to not have an appreciable effect on $T_{\boldsymbol{\tau}}$. This effect is modeled through $P(\Delta T_i|O_\Omega, n_\Omega)$, which describes how well the ego robot can negotiate around a dynamic obstacle, and may be learned offline through the simulation of many different scenarios.

Fig. 6.2 also motivates an additional simplification, in that not all locations defined by $P(O_\Omega, n_\Omega) > 0$ need to be considered; in this example, the $P(O_\Omega, n_\Omega)$ associated with corner 1 is enough to consider the effects of epistemic uncertainty of occluded dynamic obstacles, while corner 2 can be effectively ignored to reduce run time computation. With this in mind, our approach does not take the full sum over O_Ω in (6.7), but instead focuses on a single location O'_Ω:

$$P(\Delta T_i) \approx \sum_{n_\Omega} P(\Delta T_i | O'_\Omega, n_\Omega) P(O'_\Omega, n_\Omega) \tag{6.8}$$

This location is chosen to maximize the worst case ΔT_i that is possible to happen, i.e., $P(O'_\Omega, n_\Omega) > 0$ and $P(\Delta T_i | O'_\Omega, n_\Omega) > 0$. This helps reduce the number of different possibilities that need to be considered at run time, and instead focuses on the worst-case possibility.

6.3.2 Online Modelling: Inferring Probability of Dynamic Obstacle Observations

The distribution $P(O_\Omega, n_\Omega)$ describes the probability of observing n dynamic obstacles emerging from an occluded region at O. While this probability may be created offline, the proposed approach seeks to infer $P(O_\Omega, n_\Omega)$ online using historical observations of dynamic obstacles moving within the environment. This enables the framework to evaluate occlusion-aware risk not only for a particular static map $\mathcal{M}$, but also for a particular dynamic obstacle traffic pattern that has been historically observed. Likewise, if no dynamic obstacles have been observed emerging from an occluded region, then the proposed framework could determine that this occluded region has no risk of increased travel time based upon this historical evidence.

In order to model $P(O_\Omega, n_\Omega)$ online, we split this probability into the following factors:

$$P(O_\Omega, n_\Omega) = P(\Omega = 1 | O) P(O, n) \tag{6.9}$$

Here, $P(O, n)$ is the probability of n dynamic obstacles having location and velocity O within time window Δt_O, regardless of if they are emerging from an occluded area. This probability is assumed to be inherent to the environment in which the ego robot is travelling. Such a probability is affected by the size, shape and layout of the static map $\mathcal{M}$, as well as the traffic patterns of the dynamic obstacles $\mathcal{D}$. The other term $P(\Omega = 1 | O)$ describes the probability that a dynamic obstacle is observed emerging from an occluded region at O. Evaluating this factor is simple due to the line-of-sight nature of the sensors, since this can only occur at the edge of the FOV $\delta\mathcal{F}$:

$$P(\Omega = 1 | O, n) = \begin{cases} 1, & \text{if } \boldsymbol{p}_O \in \delta\mathcal{F} \\ 0, & \text{else} \end{cases} \tag{6.10}$$

This essentially restricts the sum defined in 6.7 to be along $\delta\mathcal{F}$, determined by the path segment starting location $\boldsymbol{p}_i$ and the static obstacles $\mathcal{M}$. Evaluating the factor $P(O, n)$ is a more involved process, as it requires estimating the probability of observing a dynamic obstacle at O using only historical data $\{O_i\}$.

Formally Defining Probability of Observation

The distribution $P(O, n) = P([\boldsymbol{p}_O, \boldsymbol{v}_O], n)$ is defined over $\mathbb{R}^4 \otimes \mathcal{N}$, and describes the probability of sensing n dynamic obstacles at a particular position $\boldsymbol{p}_O$ and velocity $\boldsymbol{v}_O$ within a time window t_O. The goal for the proposed approach is to infer an approximate distribution for $P(O, n)$ strictly from historical observations of the dynamic obstacles within the environment. This term is broken up into three distinct components in order to make this inference process more tractable at run time:

$$P(O, n) = P(|\boldsymbol{v}_O|)P(\hat{\boldsymbol{v}}_O|\boldsymbol{p}_O)P(n|\boldsymbol{p}_O) \tag{6.11}$$

The first component $P(|\boldsymbol{v}_O|)$ describes the probability of observing a dynamic obstacle with a certain velocity magnitude, i.e., speed. Here, the simplifying assumption is made that the speeds of all dynamic obstacles are independent of where the dynamic obstacle is located and which direction it is heading. The second component $P(\hat{\boldsymbol{v}}_O|\boldsymbol{p}_O)$ describes the probability of observing the dynamic obstacle with a certain velocity heading $\hat{\boldsymbol{v}}_O$, given a particular location. Lastly, the distribution $P(n|\boldsymbol{p}_O)$ describes the probability of observing n dynamic obstacles at a particular location over a time window Δt_O.

The advantage of breaking up $P(O, n)$ into three distinct components via (6.11) is that each one may be independently calculated from a set of observations $\{O_i\}$, and then used to calculate the full probability $P(O, n)$. Sec. 6.3.2 details how this may be exploited to effectively estimate $P(O, n)$ using these run time observations.

Estimating Probability of Observation from Historical Observations

While there are many methods of inferring a probability distribution from historical data, the proposed framework uses a parametric approach towards estimating $P(O, n)$ from $\{O_i\}$. Each component of (6.11) is assumed to take the form of an underlying distribution defined by a single parameter, and observations $\{O_i\}$ determine what values these parameters should take. Simple distributions are chosen for these estimations, as the ultimate probability $P(T_{\boldsymbol{\tau}})$ must be tenable to solve 6.1 at run time. First, $P(|\boldsymbol{v}_O|)$ is assumed to take the form of an impulse function:

$$P(|\boldsymbol{v}_O|) := \delta\left(|\boldsymbol{v}_O| - \widetilde{|\boldsymbol{v}_O|}\right) \tag{6.12}$$

Here, $|\boldsymbol{v}_O|$ describes the input variable of the impulse function and $\widetilde{|\boldsymbol{v}_O|}$ is the parameter that defines where the distribution is concentrated. As observations are collected at run time, the parameter $\widetilde{|\boldsymbol{v}_O|}$ is taken as the average of all previously observed dynamic obstacles speeds:

$$\widetilde{|\boldsymbol{v}_O|} = \frac{1}{|\{O_i\}|} \sum_{\{O_i\}} |\boldsymbol{v}_{Oi}| \tag{6.13}$$

Similarly, $P(\hat{\boldsymbol{v}}_O|\boldsymbol{p}_O)$ is also assumed to be an impulse function defined over global angle $\angle\hat{\boldsymbol{v}}_O$ determined by the heading of the unit vector $\hat{\boldsymbol{v}}_O$:

$$P(\hat{\boldsymbol{v}}_O|\boldsymbol{p}_O) = \delta\left(\angle\hat{\boldsymbol{v}}_O - \angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)\right) \tag{6.14}$$

Here, the parameter $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$ is also calculated as the average of all previously observed dynamic obstacles headings. But unlike $\widetilde{|\boldsymbol{v}_O|}$, the parameter $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$ is dependent on the location of the dynamic obstacle $\boldsymbol{p}_O$. Thus, a different average must be maintained for all possible $\boldsymbol{p}_O$, and the terms in this average are restricted to observations O_i for which $\boldsymbol{p}_{Oi} = \boldsymbol{p}_O$, denoted as $\{O_i|\boldsymbol{p}_O\}$

$$\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O) = \frac{1}{|\{O_i|\boldsymbol{p}_O\}|} \sum_{\{O_i|\boldsymbol{p}_O\}} \angle\boldsymbol{v}_{Oi} \tag{6.15}$$

Finally, $P(n|\boldsymbol{p}_O)$ is assumed to be a Poisson distribution, since it appropriately describes the occurrence of n discrete events over a finite time window Δt_O. Using the Poisson distribution is useful because it also only requires the average number of dynamic obstacles $\lambda(\boldsymbol{p}_O)$ at a given location.

Let $\lambda(\boldsymbol{p}_O)$ define the average number of dynamic obstacles at location $\boldsymbol{p}_O$ for a given environment and time window Δt_O. The Poisson distribution is defined as:

$$P(n|\boldsymbol{p}_O) = \frac{(\lambda(\boldsymbol{p}_O))^n\, e^{-\lambda(\boldsymbol{p}_O)}}{n!} \tag{6.16}$$

Similar to $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$, the parameter $\lambda(\boldsymbol{p}_O)$ must be estimated for different global positions $\boldsymbol{p}_O$ throughout the environment. Additionally, estimation of $\lambda(\boldsymbol{p}_O)$ must only use observations grouped over non-overlapping time windows of length Δt_O. The set of observations that satisfy this subset are denoted by $\{O_i|\boldsymbol{p}_O, \Delta t_O\}$.

$$\lambda_{\boldsymbol{p}_O} = \frac{1}{|\{O_i|\boldsymbol{p}_O, \Delta t_O\}|} \sum_{\{O_i|\boldsymbol{p}_O, \Delta t_O\}} n_i \tag{6.17}$$

Together, (6.13), (6.15) and (6.17) provide a means of calculating parameters from historical observations that fully define $P(O, n)$ via (6.11). Thus, $P(O, n)$ is modeled online and is flexible to different static obstacle layouts and dynamic obstacle traffic patterns. Furthermore, this model $P(O, n)$ may be used to define $P(O_\Omega, n_\Omega)$ via (6.9) and (6.10). This is one of two components required to define the distribution of travels excess travel times $P(\Delta T_i)$ defined in (6.8). Section 6.3.3 details how $P(\Delta T_i|O_\Omega, n_\Omega)$ is determined offline.

6.3.3 Offline Modelling: Learning Dependence on Tracking Time

The factor $P(\Delta T_i|O_\Omega, n_\Omega)$ acts as a prediction of how well the robot can negotiate n occluded obstacles. In order to simplify the terms in (6.7), we assume the distribution $P(\Delta T_i|O, n)$ takes the form of an impulse function:

$$P(\Delta T_i|O_\Omega, n_\Omega) = \delta\left(\Delta T_i - \widetilde{\Delta T}(O_\Omega, n_\Omega)\right) \tag{6.18}$$

The function $\widetilde{\Delta T}(O_\Omega, n_\Omega)$ is the estimate of extra time incurred from observing n dynamic obstacles emerge from behind an occlusion at O, and is modeled with a machine-learned neural network. In order to train this network, many interactions are simulated offline between an ego robot and a single dynamic obstacle ($n_\Omega = 1$) emerging from an occlusion, and the resulting travel times between different locations are recorded. These results for $n_\Omega = 1$ are generalized for multiple obstacles through the following approximation:

$$\widetilde{\Delta T}(O_\Omega, n_\Omega) \approx n_\Omega \cdot \widetilde{\Delta T}(O_\Omega, 1) \tag{6.19}$$

In other words, the excess time of n obstacles is approximately n times the excess time of observing a single previously-occluded obstacles. This helps reduce the dimension of the Monte Carlo sampling space, as well as simplify the architecture of the neural network needed to learn $\widetilde{\Delta T}(O_\Omega, 1) := \widetilde{\Delta T}(O_\Omega)$.

Fig. 6.3 shows two example simulations in which the robot is placed in an occluding environment starting from location $\boldsymbol{p}$, and is tasked with navigating towards a goal position $\boldsymbol{g}$ some distance away. The distance between start and goal is chosen to reflect a desired time window Δt_s that is large enough to effectively capture the effect of the occluded dynamic obstacle (typically $\sim$ 3-4 seconds for ground robots). A dynamic obstacle is placed along the ego robot FOV boundary $\delta\mathcal{F}(\boldsymbol{p}, \mathcal{M})$ and given its own goal to move towards. Additionally, movement and obstacle avoidance policies must be assumed for the robot and the dynamic obstacle in order to actually run the simulation. The simulation ends when the robot reaches $\boldsymbol{g}$ and the total travel time T is recorded.

In Fig. 6.3 the red starting location is close to the occluding corner, so that robot and the dynamic obstacle must suddenly avoid each other, producing sub-optimal motion and causing $\Delta T > 0$ for this simulation. The green starting location, on the other hand, is further away from the occluding corner, allowing more distance between the robot and the dynamic obstacle, resulting in near time-optimal motion and $\Delta T \approx 0$.

After many simulations are performed, the collected data are used to train a model that can infer $\Delta T_i = T_i - T_i^0$ for two consecutive points in a planned path. This model $\widetilde{\Delta T}(O_\Omega)$ uses carefully chosen features to make this inference. These features include:

- The dynamic obstacle location and velocity, the same features contained within observation O.

- The goal location $\boldsymbol{g}$.

Both of these features are transformed into the reference frame of the robot to help with the generalization and training process.

6.3.4 Graph-Search Approach For Risk-Sensitive Path Planning

Both $P(O_\Omega, n_\Omega)$ and $P(\Delta T_i|O, n)$ may be used to find $P(T_{\boldsymbol{\tau}})$, which in turn is used to calculate the risk metric in (6.1). Although there are many optional risk metrics, the proposed approach uses Conditional Value at Risk (CVaR), a well-accepted risk metric within the robotics community that is sensitive to worst-case scenarios [75]. CVaR is defined as the expected value over the worst α percent of a distribution:

$$\text{CVaR}_\alpha(T_{\boldsymbol{\tau}}) = \int_{\text{VaR}_\alpha}^{\infty} T_{\boldsymbol{\tau}} P(T_{\boldsymbol{\tau}}) dT \tag{6.20}$$

Here, the Value at Risk (VaR) is defined as the $(1-\alpha)$-quantile of a probability distribution

$$\{\text{VaR}_\alpha(T_{\boldsymbol{\tau}}) \in \mathbb{R} : F_{P(T_{\boldsymbol{\tau}})}(\text{VaR}_\alpha(T_{\boldsymbol{\tau}})) = 1 - \alpha\}, \tag{6.21}$$

where $F_{P(T_{\boldsymbol{\tau}})}(\cdot)$ represents the cumulative distribution function of $P(T_{\boldsymbol{\tau}})$. Our proposed approach thus requires an optimal path planning strategy that can find a path $\boldsymbol{\tau}$ that optimizes for risk metric $\rho = \text{CVaR}_\alpha(T_{\boldsymbol{\tau}})$.

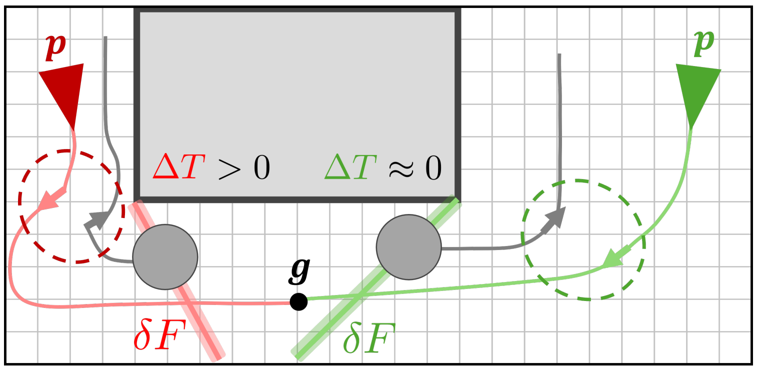


Figure 6.3: Two example simulations on which the neural network $\widetilde{\Delta T}(O_\Omega)$ is trained.

In order to find a path that optimizes $\mathrm{CVaR}_\alpha(T_{\boldsymbol{\tau}})$, let us first use (6.3) to rewrite:

$$\mathrm{CVaR}_\alpha(T_{\boldsymbol{\tau}}) = \mathrm{CVaR}_\alpha\left(\sum_i T_i\right) \tag{6.22}$$

We can then exploit the sub-addivity property of CVaR [75] to provide an upper bound on this quanity:

$$\mathrm{CVaR}_\alpha\left(\sum_i T_i\right) \leq \sum_i \mathrm{CVaR}_\alpha(T_i) \tag{6.23}$$

This relationship means that a conservative estimate for $\mathrm{CVaR}_\alpha(T_{\boldsymbol{\tau}})$ may be calculated as the sum of individual $\mathrm{CVaR}_\alpha(T_i)$ over $\boldsymbol{\tau}$. Using (6.4), we can separate out the deterministic portion so that $\mathrm{CVaR}_\alpha(T_i) = T_i^0 + \mathrm{CVaR}_\alpha(\Delta T_i)$. Lastly, given the approximation for $P(\Delta T_i)$ defined in (6.8) as well as the definitions of $P(O_\Omega, n_\Omega)$ and $P(\Delta T_i | O, n)$ detailed in Sec. 6.3.2 and Sec. 6.3.3, respectively, we can define $\mathrm{CVaR}_\alpha(T_i)$ as

$$\mathrm{CVaR}_\alpha(T_i) = T_i^0 + \mathrm{CVaR}_\alpha(n_\Omega)\widetilde{\Delta T}(O_\Omega) \tag{6.24}$$

Here, n_Ω is the stochastic number of dynamic obstacles distribution according to (6.16), and $\widetilde{\Delta T}(O_\Omega)$ is the neural network trained from offline simulations. The value of $\mathrm{CVaR}_\alpha(T_i)$ may be treated as an edge cost for a graph-search algorithm in which the total path cost is defined by (6.22), whose upper bound is defined via the sum of all edge costs (6.23). Any graph search algorithm may then be used to find the globally optimal path that optimizes this risk metric.

We close this section by noting particular advantages of using (6.24) as an edge cost. First, the product $\mathrm{CVaR}_\alpha(n_\Omega)\widetilde{\Delta T}(O_\Omega)$ represents the inferred risk of incurring excess time due to occluded dynamic obstacles. If $\mathrm{CVaR}_\alpha(n_\Omega)\widetilde{\Delta T}(O_\Omega) \approx 0$, e.g., there are no dynamic obstacles within the environment that may be occluded, then $\mathrm{CVaR}_\alpha(T_i) = T_i^0$ and the resulting graph search would return the standard minimum-time path. If $\mathrm{CVaR}_\alpha(n_\Omega)\widetilde{\Delta T}(O_\Omega) > 0$, then there are two possible mechanism which may lead to a higher edge cost:

- A given environment may be trafficked with a high number dynamic obstacles, raising $\mathrm{CVaR}_\alpha(n_\Omega)$.

- A single dynamic obstacle causes a large excess time, raising $\widetilde{\Delta T}(O_\Omega)$.

6.4 Preliminary Simulation Results

This section presents preliminary results for the proposed approach. First, Sec. 6.4.1 discusses the process for collecting data from simulation for the offline-trained inference model. Sec. 6.4.2

then discusses how run time observations of dynamic obstacles are used towards the online-created inference model. Finally, Sec. 6.4.3 shows these components working within a full autonomy stack for a high-fidelity simulator.

6.4.1 Data Collection and Offline Trained Model

A simulator was used to collect data for training $\widetilde{\Delta T}(O_\Omega)$, determining how much excess time ΔT_i may be incurred when a dynamic obstacle emerged from an occluded region at O. This simulator was designed to be as lightweight as possible to quickly collect data. For accurate training, the simulator must realistically reflect both the ego robot model and control policy to avoid obstacles while moving towards a goal, as well as the model and motion policy of the dynamic obstacles moving through the environment. For the ego robot, a unicycle model [48] was assumed with state $\boldsymbol{x} = [p_x, p_y, \theta]^T$ defined by global Cartesian coordinates (p_x, p_y), heading θ, and controls $\boldsymbol{u} = [v, \omega]^T$. A DWA controller [33] that ran at 50 Hz was used to determine $\boldsymbol{u} = [v, \omega]^T$, with $v \in [-0.2, 0.8]$ m/s and $\omega \in [-2.0, 2.0]$ rad/s. For the dynamic obstacles, an empirically-derived social force model [59] was used to realistically simulate how a human may try to negotiate around the ego robot.

A simple rectangular environment was used to simulate interactions between the robot and a dynamic obstacle emerging from an occluded region. In each simulation, the robot was initially placed at a random unoccupied position $\boldsymbol{p}_0$ and heading with a goal also chosen in a random unoccupied position $\boldsymbol{g}$. From this position, the dynamic obstacle was placed at a random position along the FOV edge $\delta\mathcal{F}$, and itself given a random goal so that it would emerge from the occlusion (rather than going further into the occluded region). With these initial conditions set, the simulation was ran until the ego robot reached its goal position. This selection method allowed the data to cover a wide set of possible scenarios the robot may encounter, including different possible relative positions and headings for the dynamic obstacle emerging from an occluded region. For each simulation, several pieces of information were recorded:

- $\boldsymbol{g} \in \mathbb{R}^2$: The goal position
- $\boldsymbol{p}_\Omega \in \mathbb{R}^2$: The position of the emerging dynamic obstacle
- $\boldsymbol{v}_\Omega \in \mathbb{R}^2$: The velocity of the emerging dynamic obstacle
- T: The time taken for the robot to travel from $\boldsymbol{p}_0$ to $\boldsymbol{g}$

The vectors $\boldsymbol{g}$, $\boldsymbol{p}$ and $\boldsymbol{v}$ were all measured in the robot reference frame. Fig. 6.4 shows this an actual example simulation between the robot control policy and the dynamic obstacle motion policy. The excess time ΔT was calculated by first determining the nominal time to reach the goal T_0 (reasonably approximated via $T_0 \approx |\boldsymbol{g} - \boldsymbol{p}_0|/v_{\max}$) and taking the difference $\Delta T = T - T_0$. For the example shown in Fig. 6.4, the excess was calculated to be $\Delta T = 4.2$ s since the dynamic obstacle emerged from the occlusion so close to the robot.

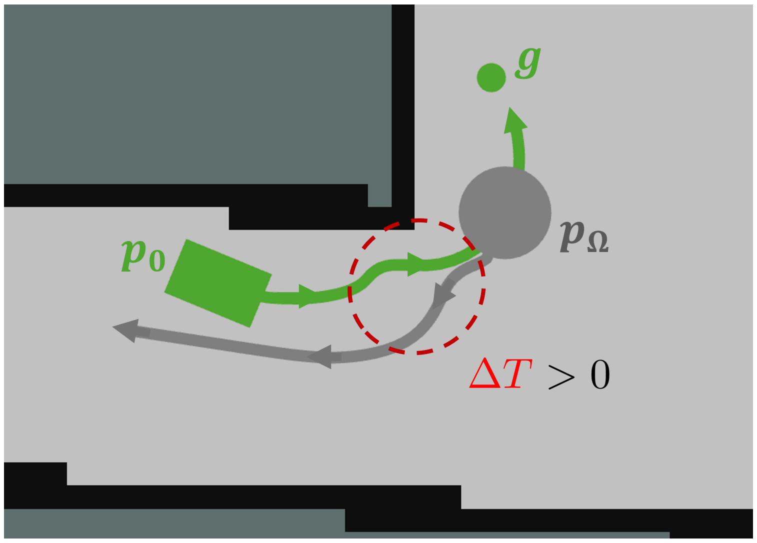


Figure 6.4: Actual simulation between robot and dynamic obstacle.

Data were collected from 5×10^4 simulations, recording $\{\boldsymbol{g}, \boldsymbol{p}_\Omega, \boldsymbol{v}_\Omega, \Delta T\}$ for each. This data was used to train a model $\widetilde{\Delta T}(O_\Omega)$ to predict ΔT from $O_\Omega = [\boldsymbol{p}_\Omega, \boldsymbol{v}_\Omega]$, as well as the goal location $\boldsymbol{g}$. The model for $\widetilde{\Delta T}$ was chosen to be a five-layer neural network, with 6 and 1 as the input and output layer dimension, respectively, and $[8, 6, 4]$ as the hidden layer dimensions. Once trained, the model $\widetilde{\Delta T}(O_\Omega)$ was used as a parameter that defined $P(\Delta T_i | O_\Omega, n_\Omega)$ via 6.18. In particular, this model was used in 6.24 to directly predict of how well the robot can negotiate an occluded obstacle observed at O_Ω. This offline-trained component thus informed how much the excess travel time ΔT would be impacted by a dynamic obstacle emerging from a particular occlusion.

6.4.2 Inferring Distribution of Observations at Run Time

The second component needed for the proposed approach is the online-created model $P(O, n)$ that describes the probability of n dynamic obstacles being observed at a particular position and velocity. This probability is split into three distinct factors via (6.11), where each factor describes (i) the distribution of speeds for a dynamic obstacles (ii) the distribution of dynamic obstacle headings and (iii) the number of dynamic obstacles observed at location $\boldsymbol{p}_O$. Each component is dependent on a single parameter defined by the average value for each component:

- $\widetilde{|\boldsymbol{v}_O|}$, defined in (6.13) as the average historical speed of dynamic obstacles
- $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$, defined in (6.15) as the average historical heading angle of dynamic obstacles observed at position $\boldsymbol{p}_O$

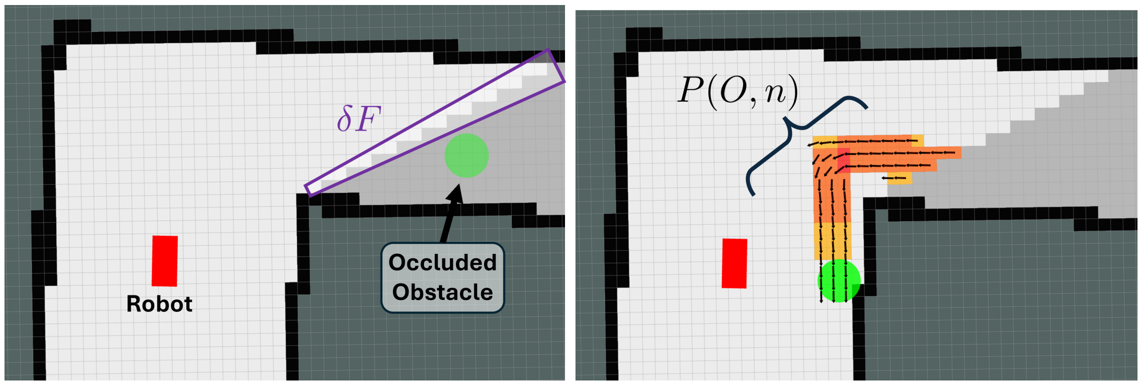


Figure 6.5: Actual simulation in which parameters for $P(O,n)$ are estimated from run time observations.

- $\lambda(\boldsymbol{p}_O)$, defined in (6.17) as the historical average number of dynamic obstacles observed at position $\boldsymbol{p}_O$

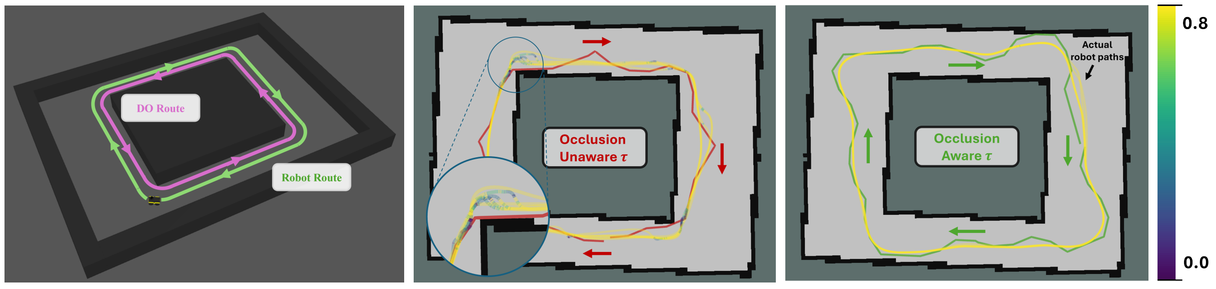
Note that $|\widetilde{\boldsymbol{v}_O}|$ is relatively easy to calculate, since it is just an average over all dynamic obstacle speeds observed. However, both $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$ and $\lambda(\boldsymbol{p}_O)$ are averages that are functions of position $\boldsymbol{p}_O$, which is an uncountably infinite input space. To make these parameters tractable to estimate at run time, a discrete estimate $\mathcal{M}_\mathrm{d} \in \mathbb{R}^2$ is used in place of the true continuous global map $\mathcal{M} \in \mathbb{R}^2$, where each position $\boldsymbol{p}_\mathrm{d} \in \mathcal{M}_d$ represents a small grid cell of the position space, centered on $\boldsymbol{p}_\mathrm{d}$ and sized 0.2 m on each side.

Fig. 6.5 shows a real simulation where these parameters are estimated at run time. Fig. 6.5(a) shows how initially the robot has not observed dynamic obstacles within its FOV, and so $P(n = 0|\boldsymbol{p}_O) = 1$ for these locations. A time window of $\Delta t_O = 1$ s is defined, and dynamic obstacle observations $\{O_i\}$ are are collected within this window. Fig. 6.5(b) shows what this portion of the map looks like after the dynamic obstacle has entered within the FOV for some time. The arrows indicate the learned dynamic obstacle heading $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$ at each grid location, and orange colored grid cells indicate a learned $\lambda(\boldsymbol{p}_O)$.

Estimating these parameters online allows the construction of two critical pieces of the proposed approach: first, this allows an estimate of $\mathrm{CVaR}_\alpha(n_\Omega)$, i.e. a risk metric on the number of obstacles expected at a given location $\boldsymbol{p}_O$. Second, if there is a non-zero probability of observing a dynamic obstacle at a given location $\boldsymbol{p}_O$, $|\widetilde{\boldsymbol{v}_O}|$ and $\angle\widetilde{\hat{\boldsymbol{v}}_O}(\boldsymbol{p}_O)$ may be used to estimate the velocity $\boldsymbol{v}_O$ of such an observation. Both $\boldsymbol{p}_O$ and $\boldsymbol{v}_O$ are then used to infer excess time via the offline-trained model $\widetilde{\Delta T}(O_\Omega)$. These are used as a means of estimating the risk of moving within an occluded environment, and ultimately are combined to inform the creation of an optimal path that reduces this risk.

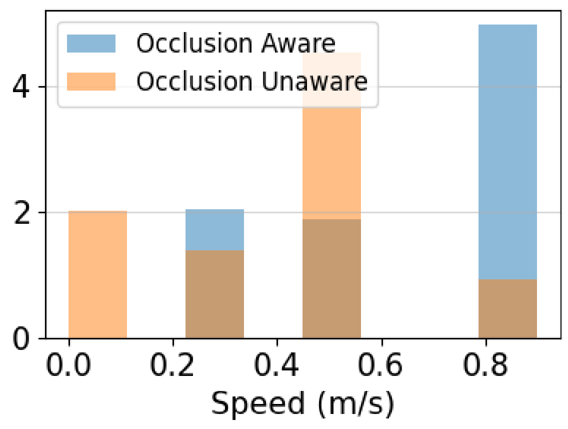
6.4.3 Example Within Full Autonomy Stack

The proposed occlusion-aware path planning framework was implemented as part of a full autonomy stack and tested within a high-fidelity simulator. Gazebo was used to simulate a real-world Clearpath Robotics Jackal ground robot platform equipped with a 360° lidar to produce laser scans of the immediate surroundings. The SLAM package Gmapping [67] was used to created a map of static portions of the environment, while AMCL [3] was used to localize within these maps. Fig. 6.6(a) shows the gazebo environment in which the robot was placed, tasked with moving clockwise around a central rectangle. Dynamic obstacles were also placed within this environment, tasked with moving counter clockwise around, i.e., directly opposed to the robot. The proposed occlusion-aware path planner was used to generate a path $\boldsymbol{\tau}$ for which a DWA controller would track while avoiding obstacles, both static and dynamic. As the robot moved throughout the environment, dynamic obstacles were observed moving counter clockwise through the environment, which were used to infer dynamic obstacle distribution, as described in Sec. 6.4.2. This was used in conjuction with the offline-trained model, described in Sec.6.4.1, to generate an occlusion-aware path around this environment. Fig. 6.6(b) first shows the results of an occlusion-unaware path planning policy. Shown in red is the path $\boldsymbol{\tau}$ created for the robot to track, moving clockwise around the environment. Also plotted is the robot path over five minutes of traveling around this environment, colored using the commanded speed provided by the DWA. A yellor color indicates that the DWA commanded the maximum speed of 0.8 m/s, while darker colors indicate slower speeds or full stops. Fig. 6.6(b) shows how slower speeds are commanded around the occluding corners of the center rectangle, caused by dynamic obstacle emerging from the behind these occluding corners while the robot was nearing them. Fig. 6.6(c) instead shows the full approach in which an occlusion-aware path (green line) is planned for the DWA controller to track. This occlusion-aware is generated as more observations of dynamic obstacles in the environment are obtained by the robot, producing motion that increases visibility around the occluding corners. This benefits the ego robot, as Fig. 6.6(c) shows how the maximum speed is commanded over nearly the entire five minutes of simulation.

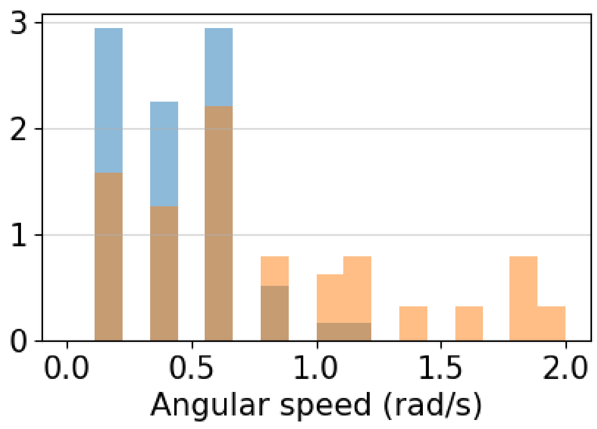


(a) (b) (c)

Figure 6.6: Simulation results for a simple rectangular environment.



(a)



(b)

Figure 6.7: Normalized histograms showing how the occlusion-aware path allowed for more frequency forward full speed and forward motion.

Additionally, Fig. 6.7(a) shows a normalized histogram of forward speeds $|v|$ commanded by the DWA planner over ten minutes of simulation. It can be seen how the occlusion-aware navigation stack resulted in higher overall frequency of the highest allowable commanded speed of 0.8 m/s. This a direct result of the improved visibility of dynamic obstacles around occlusions, as this prevented instances where the robot suddenly sees a dynamic obstacle close by, and must rapidly stop in order to negotiate around these obstacles. Additionally, Fig. 6.7(b) shows a normalized histogram of the commanded robot angular velocity ω, which tells a similar story: with occlusion-aware navigation, the robot moves along a straight path ($\omega \approx 0$) with a higher frequency when compared to occlusion-unaware navigation since it does not need to rapidly move out of the way of newly-observed dynamic obstacles.

6.5 Discussion and Conclusion

This chapter proposes an occlusion-aware path planning policy created using (i) a model trained offline to predict how well a robot motion policy may negotiate and suddenly unoccluded dynamic obstacle, and (ii) a model created online from run time dynamic obstacle observations to accurately apply this model to the surrounding environment. Together, both models are used to infer a risk of increased travel time due to the epistemic uncertainty caused by potentially occluded dynamic obstacles in the environment. Finally, a graph-search algorithm was used to find a path that globally minimizes such risk. The resulting approach to occlusion-aware navigation promises both expressive power and flexibility towards new and unknown environments.

The proposed approach relies on many approximations in order to tractably estimate risk at run time, but these simplifications may not behave well in all environments. For example, this work models the predicted heading and speed of a recently unoccluded dynamic obstacle as a uni-model impulse function, but the true underlying distribution may be best described using multiple modes (e.g., humans and service AMR moving in the same space may have different operating speeds). This simplification may hide certain risks associated with a given occlusion, which may impact the overall performance of the framework. Thus, one future research direction may be to consider multimodal distributions when modeling the probability distribution of dynamic obstacles within the environment. Additionally, creating the offline model discussed in this approach required simulations and pre-training of a neural network. In order to perform these simulations, a motion policy for the dynamic obstacles must be assumed a priori. One future extension of this work would be to account for additional uncertainty in the motion policy of the dynamic obstacle, using historic observations to predict not only where dynamic obstacles may emerge from occlusions, but also how they might behave.

Lastly, the simulation results discussed in Sec. 6.4 used a DWA controller to track a planned path $\boldsymbol{\tau}$ while avoiding dynamic obstacles. However, other such policies exist that attempt to negotiate dynamic obstacles in different ways, and although DWA is still a commonly adopted approach for AMR, it may be worth investigating how different robot control policies may effect ΔT_i. This would ultimately effect the path planned by this approach, as it may be that cutting close to the corner is actually advantageous with other robot control policies. Thus, the behavior and efficacy of the proposed framework may be affected by motion policy of the robot, and should be explored in future work.

Part IV

Epilogue

Chapter 7

Conclusions and Future Work

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 possible directions we could take for future work to build on what we have achieved thus far.

7.1 Conclusions

In this dissertation, various frameworks for occlusion-aware navigation have been presented. Part I discussed two different approaches towards visibility-aware motion that could balance between increasing visibility around occlusions while moving towards a goal. This was accomplished through the development of analytically simple yet expressive perception objective that, when included as a component of the cost function of an OCP, served to promote motion that reduced the geometric area occluded to the travelling robot. Such a cost component could be quickly deployed within any typical navigation stack with no pre-training required. Additionally, safety modules were included that regulated speed and direction of travel as a means of ensuring obstacle avoidance. Together, it was shown how the complete framework could produce trajectories that improved visibility around occlusions, and because of this allowed faster commanded speeds over the entire trajectory.

Beyond visibility-aware approaches towards occlusion-aware navigation, a more sophisticated treatment involves establishing a risk metric that relates to the uncertainty associated with the occluded region and attempting to produce a path or trajectory that minimizes this risk metric. To this end, Part II describes a risk-aware trajectory generation technique that associates trajectories to a certain risk of collision. This discussion establishes important concepts used in susbsequent Part III, such as the utilization of the risk-sensitive VaR instead of the typical expected value, or an emphasis on a data-informed risk metric that requires relatively little data to train compared to other learning-based techniques to estimate risk. The proposed trajectory generation approach could leverage this data-informed risk to adapt the planned trajectory to the surrounding environment, commanding slower speeds to reduce the risk of collision from possible trajectory tracking error.

Finally, Part III discusses how risk-aware navigation may be applied towards the occlusion-aware navigation problem. The first approach discussed leverages a pre-existing occlusion unaware planner as the initial navigation policy, and efficiently uses data collected at run time to adjust this policy on the fly, allowing the policy to adapt to the environment in which the robot is deployed. The collected data helps inform a risk-metric created through the QTD algorithm and associates a given trajectory with a certain risk of entering into an undesired set. This risk metric is included inside the cost function of a graph-search algorithm, and the resulting approach is shown through simulation and experiment to improve motion not only for the ego robot but also for other dynamic obstacle traveling in the same environment. Preliminary discussion is also included on a second approach that focuses on inferring the risk of a negative outcome from historical observation of the dynamic obstacles within the environment, rather than direct experience of those negative outcomes by the ego robot. This approach attempts to quantify the effect of uncertainty associated with occluded dynamic obstacles on the total time to track a path, then establishes the estimate of a risk metric over a planned path. This allows the total risk inference to be decomposed into two distinct models, one that is trained with offline simulated data and one that is created online with dynamic obstacle data collected at run time. This preliminary approach shows promising results for a framework that can adapt to a given environment and create a policy bespoke to the surrounding risk specific to that environment.

7.2 Discussion and Possible Future Work

One commonality throughout the work presented in this dissertation is the focus on the creation of a cost function that, when optimized over a planned path, resulted in the desired occlusion-aware behavior that sought to promote visibility and reduce uncertainty. Part I focused on the creation of an analytical approximation of occluded area, while Part II and Part III explored using a data-informed risk metric inside the cost function. Thus, one way to view this work is that of cost/reward engineering for an optimal control problem, which comes with its own benefits and challenges. In one sense, relying on a single utility function to determine behavior is the definition of rational behavior [7], and any autonomy policy must be rational and explainable if we expect it to be deployed in the real world. In another sense, using a scalar number to judge and choose the best value for a decision variable has practical value, as there are many techniques to optimize over a decision variable. The challenge, however, is to create a cost function that can (i) be optimized over at run time while (ii) encoding potentially complex behaviors and produce policies that mimic or out-perform humans. This is a subject that has been considered in both multi-objective optimization [52] and reinforcement learning [37]. This dissertation is included in a larger effort to explore different ways of crafting a cost function that accomplishes these desiderata while remaining tractable for a computer to execute on board a robot at run time. Furthermore, there is currently a gap between complex approaches that require millions of training data that achieve human-level performance, and simpler approaches that are easier to implement and interpret but cannot achieve a high level of performance. The work presented in Part II and Part III attempts to bridge this gap by exploring techniques that can learn much smaller amounts of data (tens to hundreds) by adopting simpler learning models as components within more classical planning approaches. Part III extends this idea even further by using data collected at run time to inform the risk-sensitive path planning policy.

As development continues on improving AMR toward human and superhuman capability, there is clear need for path planning policies to account for the uncertainty created by occlusions. While this dissertation details effort towards this goal, there is still much left to explore. One potential research direction would be an exploration in more nuanced ways of defining the undesired set of Chapter 5. While this work considers visibility of unoccluded obstacles, there may be other ways to define this undesired set; for example, such a framework could lend well towards the study of human-robot interaction, avoiding potentially negative or “awkward” interactions between robot and humans when rounding occluding corners. The undesired set could include all negative interactions between robot and humans, and learn there is a probability of these interactions occurring around occlusions. Another potential research direction is investigation of the framework presented in Chapter 6 for further analytical performance guarantees, ways to relax the simplifying assumptions, or a practical means of finding an optimal path over risk metric.

Bibliography

[1] Mohamadreza Ahmadi, Ugo Rosolia, Michel D. Ingham, Richard M. Murray, and Aaron D. Ames. “Risk-Averse Decision Making Under Uncertainty”. In: *IEEE Transactions on Automatic Control* 69.1 (2024), pp. 55–68. DOI: `10.1109/TAC.2023.3264178`.

[2] Matthias Althoff and Silvia Magdici. “Set-based prediction of traffic participants on arbitrary road networks”. In: *IEEE Transactions on Intelligent Vehicles* 1.2 (2016), pp. 187–202.

[3] *AMCL*. `http://wiki.ros.org/amcl`. Accessed: 2024-02-22.

[4] Hans Andersen, Javier Alonso-Mora, You Hong Eng, Daniela Rus, and Marcelo H. Ang. “Trajectory Optimization and Situational Analysis Framework for Autonomous Overtaking With Visibility Maximization”. In: *IEEE Transactions on Intelligent Vehicles* 5.1 (2020), pp. 7–20. DOI: `10.1109/TIV.2019.2955361`.

[5] Hans Andersen, Wilko Schwarting, Felix Naser, You Hong Eng, Marcelo H. Ang, Daniela Rus, and Javier Alonso-Mora. “Trajectory optimization for autonomous overtaking with visibility maximization”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–8. DOI: `10.1109/ITSC.2017.8317853`.

[6] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[7] Kenneth J Arrow. “Economic theory and the hypothesis of rationality”. In: *Utility and Probability*. Springer, 1990, pp. 25–37.

[8] Marc G Bellemare, Will Dabney, and Rémi Munos. “A distributional perspective on reinforcement learning”. In: *International conference on machine learning*. PMLR. 2017, pp. 449–458.

[9] Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional reinforcement learning*. MIT Press, 2023.

[10] Lars Blackmore, Masahiro Ono, and Brian C Williams. “Chance-constrained optimal path planning with obstacles”. In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094.

[11] Amanda Bouman, Muhammad Fadhil Ginting, Nikhilesh Alatur, Matteo Palieri, David D Fan, Thomas Touma, Torkom Pailevanian, Sung-Kyun Kim, Kyohei Otsu, Joel Burdick, et al. “Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2518–2525.

[12] Amanda Bouman et al. “Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2518–2525. DOI: 10.1109/IROS45743.2020.9341361.

[13] Maxime Bouton, Alireza Nakhaei, Kikuo Fujimura, and Mykel J. Kochenderfer. “Scalable Decision Making with Sensor Occlusions for Autonomous Driving”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2076–2081. DOI: 10.1109/ICRA.2018.8460914.

[14] Anthony Brunel, Amine Bourki, Cédric Demonceaux, and Olivier Strauss. “Splatplanner: Efficient autonomous exploration via permutohedral frontier filtering”. In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 608–615.

[15] Adam Bry and Nicholas Roy. “Rapidly-exploring random belief trees for motion planning under uncertainty”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 723–730.

[16] Andrea Camisa, Andrea Testa, and Giuseppe Notarstefano. “Multi-Robot Pickup and Delivery via Distributed Resource Allocation”. In: *IEEE Transactions on Robotics* 39.2 (2023), pp. 1106–1118. DOI: 10.1109/TRO.2022.3216801.

[17] Svante Carlsson, Håkan Jonsson, and Bengt J Nilsson. “Finding the shortest watchman route in a simple polygon”. In: *Discrete & Computational Geometry* 22 (1999), pp. 377–402.

[18] Mateo Guaman Castro, Samuel Triest, Wenshan Wang, Jason M. Gregory, Felix Sanchez, John G. Rogers, and Sebastian Scherer. “How Does It Feel? Self-Supervised Costmap Learning for Off-Road Vehicle Traversability”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 931–938. DOI: 10.1109/ICRA48891.2023.10160856.

[19] Yuan Chang, Han Zhou, Xiangke Wang, Lincheng Shen, and Tianjiang Hu. “Cross-Drone Binocular Coordination for Ground Moving Target Tracking in Occlusion-Rich Scenarios”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3161–3168. DOI: 10.1109/LRA.2020.2975713.

[20] Kuo Chen, Jingang Yi, and Dezhen Song. "Gaussian Processes Model-Based Control of Underactuated Balance Robots". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 4458–4464. DOI: `10.1109/ICRA.2019.8794097`.

[21] Kuo Chen, Jingang Yi, and Dezhen Song. "Gaussian-Process-Based Control of Underactuated Balance Robots With Guaranteed Performance". In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 572–589. DOI: `10.1109/TRO.2022.3203625`.

[22] Wei-pang Chin and Simeon Ntafos. "Optimum watchman routes". In: *Information Processing Letters* 28.1 (1988), pp. 39–44. ISSN: 0020-0190. DOI: `https://doi.org/10.1016/0020-0190(88)90141-X`. URL: `https://www.sciencedirect.com/science/article/pii/002001908890141X`.

[23] Moorad Choudhry. *An introduction to value-at-risk*. John Wiley & Sons, 2013.

[24] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. "Distributional reinforcement learning with quantile regression". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[25] Tung Dang, Christos Papachristos, and Kostas Alexis. "Autonomous exploration and simultaneous object search using aerial robots". In: *2018 IEEE Aerospace Conference*. IEEE. 2018, pp. 1–7.

[26] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. "Practical search techniques in path planning for autonomous driving". In: *Ann Arbor* 1001.48105 (2008), pp. 18–80.

[27] Noel E Du Toit and Joel W Burdick. "Robot motion planning in dynamic, uncertain environments". In: *IEEE Transactions on Robotics* 28.1 (2011), pp. 101–115.

[28] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–8. DOI: `10.1109/IROS.2018.8593739`.

[29] David D. Fan, Ali-akbar Agha-mohammadi, and Evangelos A. Theodorou. "Learning Risk-Aware Costmaps for Traversability in Challenging Environments". In: *IEEE Robotics and Automation Letters* 7.1 (2022), pp. 279–286. DOI: `10.1109/LRA.2021.3125047`.

[30] Wenhao Feng, Liang Ding, Ruyi Zhou, Chongfu Xu, Huaiguang Yang, Haibo Gao, Guangjun Liu, and Zongquan Deng. "Learning-Based End-to-End Navigation for Planetary Rovers Considering Non-Geometric Hazards". In: *IEEE Robotics and Automation Letters* 8.7 (2023), pp. 4084–4091. DOI: `10.1109/LRA.2023.3281261`.

[31] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. "qpOASES: A parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6 (2014), pp. 327–363.

[32] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. "qpOASES: a parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363. ISSN: 18672957. DOI: 10.1007/s12532-014-0071-1.

[33] 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.

[34] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. "Robot Operating System (ROS): The Complete Reference (Volume 1)". In: ed. by Anis Koubaa. Cham: Springer International Publishing, 2016. Chap. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. URL: http://dx.doi.org/10.1007/978-3-319-26054-9_23.

[35] Barry Gilhuly, Armin Sadeghi, and Stephen L. Smith. "Estimating Visibility From Alternate Perspectives for Motion Planning With Occlusions". In: *IEEE Robotics and Automation Letters* 9.6 (2024), pp. 5583–5590. DOI: 10.1109/LRA.2024.3396056.

[36] Barry Gilhuly, Armin Sadeghi, Peyman Yedmellat, Kasra Rezaee, and Stephen L. Smith. "Looking for Trouble: Informative Planning for Safe Trajectories with Occlusions". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 8985–8991. DOI: 10.1109/ICRA46639.2022.9811994.

[37] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. "Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15281–15295.

[38] Astghik Hakobyan, Gyeong Chan Kim, and Insoon Yang. "Risk-Aware Motion Planning and Control Using CVaR-Constrained Optimization". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3924–3931. DOI: 10.1109/LRA.2019.2929980.

[39] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. "ACADO toolkit - An open-source framework for automatic control and dynamic optimization". In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312. ISSN: 01432087. DOI: 10.1002/oca.939.

[40] Constantin Hubmann, Nils Quetschlich, Jens Schulz, Julian Bernhard, Daniel Althoff, and Christoph Stiller. "A POMDP Maneuver Planner For Occlusions in Urban Scenarios". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2172–2179. DOI: 10.1109/IVS.2019.8814179.

[41] Heejin Jeong, Hamed Hassani, Manfred Morari, Daniel D. Lee, and George J. Pappas. “Deep Reinforcement Learning for Active Target Tracking”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1825–1831. DOI: 10.1109/ICRA48506.2021.9561258.

[42] Heejin Jeong, Brent Schlotfeldt, Hamed Hassani, Manfred Morari, Daniel D. Lee, and George J. Pappas. “Learning Q-network for Active Information Acquisition”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 6822–6827. DOI: 10.1109/IROS40897.2019.8968173.

[43] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[44] Matthew Kelly. “An introduction to trajectory optimization: How to do your own direct collocation”. In: *SIAM Review* 59.4 (2017), pp. 849–904.

[45] Markus Koschi and Matthias Althoff. “Set-based prediction of traffic participants considering occlusions and traffic rules”. In: *IEEE Transactions on Intelligent Vehicles* 6.2 (2020), pp. 249–265.

[46] Hanna Kurniawati. “Partially observable markov decision processes and robotics”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2022), pp. 253–277.

[47] Woong Kwon, Jun Ho Park, Minsu Lee, Jongbeom Her, Sang-Hyeon Kim, and Ja-Won Seo. “Robust Autonomous Navigation of Unmanned Aerial Vehicles (UAVs) for Warehouses’ Inventory Application”. In: *IEEE Robotics and Automation Letters* 5.1 (2020), pp. 243–249. DOI: 10.1109/LRA.2019.2955003.

[48] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[49] Keuntaek Lee, Jason Gibson, and Evangelos A. Theodorou. “Aggressive Perception-Aware Navigation Using Deep Optical Flow Dynamics and PixelMPC”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1207–1214. DOI: 10.1109/LRA.2020.2965911.

[50] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. “Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3)”. In: *arXiv preprint arXiv:1003.2005* (2010).

[51] Thomas Lew, Riccardo Bonalli, and Marco Pavone. “Risk-Averse Trajectory Optimization via Sample Average Approximation”. In: *IEEE Robotics and Automation Letters* 9.2 (2024), pp. 1500–1507. DOI: 10.1109/LRA.2023.3331889.

[52] Jian-Yu Li, Zhi-Hui Zhan, Yun Li, and Jun Zhang. “Multiple tasks for multiple objectives: A new multiobjective optimization method via multitask optimization”. In: *IEEE Transactions on Evolutionary Computation* (2023).

[53] Anirudha Majumdar and Marco Pavone. “How should a robot assess risk? towards an axiomatic theory of risk in robotics”. In: *Robotics Research: The 18th International Symposium ISRR*. Springer. 2020, pp. 75–84.

[54] Gonçalo S Martins, Rui P Rocha, Fernando J Pais, and Paulo Menezes. “Clusternav: Learning-based robust navigation operating in cluttered environments”. In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 9624–9630.

[55] Courtney McBeth, James Motes, Diane Uwacu, Marco Morales, and Nancy M. Amato. “Scalable Multi-Robot Motion Planning for Congested Environments With Topological Guidance”. In: *IEEE Robotics and Automation Letters* 8.11 (2023), pp. 6867–6874. DOI: 10.1109/LRA.2023.3312980.

[56] Joseph SB Mitchell. “Approximating watchman routes”. In: *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2013, pp. 844–855.

[57] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[58] Nicholas Mohammad and Nicola Bezzo. “A Robust and Fast Occlusion-based Frontier Method for Autonomous Navigation in Unknown Cluttered Environments”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 6324–6331. DOI: 10.1109/IROS47612.2022.9982059.

[59] Mehdi Moussaid, Dirk Helbing, Simon Garnier, Anders Johansson, Maud Combe, and Guy Theraulaz. “Experimental study of the behavioural mechanisms underlying self-organization in human crowds”. In: *Proceedings of the Royal Society B: Biological Sciences* 276.1668 (2009), pp. 2755–2762.

[60] Javier Muñoz, Peter Lehner, Luis E. Moreno, Alin Albu-Schäffer, and Máximo A. Roa. “CollisionGP: Gaussian Process-Based Collision Checking for Robot Motion Planning”. In: *IEEE Robotics and Automation Letters* 8.7 (2023), pp. 4036–4043. DOI: 10.1109/LRA.2023.3280820.

[61] Varun Murali, Igor Spasojevic, Winter Guerra, and Sertac Karaman. “Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness”. In: *2019 American Control Conference (ACC)*. 2019, pp. 3936–3943. DOI: 10.23919/ACC.2019.8814697.

[62] Yannik Nager, Andrea Censi, and Emilio Frazzoli. “What lies in the shadows? Safe and computation-aware motion planning for autonomous vehicles using intent-aware dynamic shadow regions”. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5800–5806. DOI: 10.1109/ICRA.2019.8793557.

[63] Andre Nuñez, Felix H. Kong, Alberto González-Cantos, and Robert Fitch. “Risk-Aware Stochastic Ship Routing Using Conditional Value-at-Risk”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 10543–10550. DOI: 10.1109/IROS55552.2023.10341431.

[64] *Obstacle Detection ROS Package*. https://github.com/tysik/obstacle_detector. Accessed: 2024-02-22.

[65] Billy Okal and Kai O Arras. “Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning”. In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 2889–2895.

[66] Masahiro Ono, Marco Pavone, Yoshiaki Kuwata, and J Balaram. “Chance-constrained dynamic programming with application to risk-aware robotic space exploration”. In: *Autonomous Robots* 39 (2015), pp. 555–571.

[67] *OpenSLAM Gmapping*. http://wiki.ros.org/gmapping. Accessed: 2024-02-22.

[68] Piotr F Orzechowski, Annika Meyer, and Martin Lauer. “Tackling occlusions & limited sensor range with set-based safety verification”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1729–1736.

[69] Piotr F. Orzechowski, Annika Meyer, and Martin Lauer. “Tackling Occlusions and Limited Sensor Range with Set-based Safety Verification”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1729–1736. DOI: 10.1109/ITSC.2018.8569332.

[70] Sooho Park, Yu Huang, Chun Fan Goh, and Kenji Shimada. “Robot Model Learning with Gaussian Process Mixture Model”. In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. 2018, pp. 1263–1268. DOI: 10.1109/COASE.2018.8560452.

[71] Bryan Penin, Paolo Robuffo Giordano, and François Chaumette. “Vision-Based Reactive Planning for Aggressive Target Tracking While Avoiding Collisions and Occlusions”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3725–3732. DOI: 10.1109/LRA.2018.2856526.

[72] James A. Preiss, Wolfgang Honig, Gaurav S. Sukhatme, and Nora Ayanian. “Crazyswarm: A large nano-quadcopter swarm”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 3299–3304. DOI: 10.1109/ICRA.2017.7989376.

[73] Shuanghu Qiao, Kai Zheng, and Guofeng Wang. "A Path Planning Method for Autonomous Ships Based on SVM". In: *2020 Chinese Control And Decision Conference (CCDC)*. 2020, pp. 3068–3072. DOI: 10.1109/CCDC49329.2020.9164806.

[74] Victor Massagué Respall, Dmitry Devitt, Roman Fedorenko, and Alexandr Klimchik. "Fast sampling-based next-best-view exploration algorithm for a MAV". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 89–95.

[75] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of risk* 2 (2000), pp. 21–42.

[76] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[77] Markus Schratter, Maxime Bouton, Mykel J Kochenderfer, and Daniel Watzenig. "Pedestrian collision avoidance system for scenarios with occlusions". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1054–1060.

[78] *Spot CORE payload reference*. https://support.bostondynamics.com/s/article/Spot-CORE-payload-reference. Accessed: 2024-02-22.

[79] Zachary Sunberg and Mykel J. Kochenderfer. "Improving Automated Driving Through POMDP Planning With Human Internal States". In: *IEEE Transactions on Intelligent Transportation Systems* 23.11 (2022), pp. 20073–20083. DOI: 10.1109/TITS.2022.3182687.

[80] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[81] Lei Tai, Giuseppe Paolo, and Ming Liu. "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 31–36. DOI: 10.1109/IROS.2017.8202134.

[82] Rahul Tallamraju, Eric Price, Roman Ludwig, Kamalakar Karlapalem, Heinrich H. Bülthoff, Michael J. Black, and Aamir Ahmad. "Active Perception Based Formation Control for Multiple Aerial Vehicles". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4491–4498. DOI: 10.1109/LRA.2019.2932570.

[83] Sebastion Thrun. *Probabilistic Robotics*. MIT Press, 2005, p. 221.

[84] Jesus Tordesillas and Jonathan P. How. "Deep-PANTHER: Learning-Based Perception-Aware Trajectory Planner in Dynamic Environments". In: *IEEE Robotics and Automation Letters* 8.3 (2023), pp. 1399–1406. DOI: 10.1109/LRA.2023.3235678.

[85] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. "acados – a modular open-source framework for fast embedded optimal control". In: *Mathematical Programming Computation* (Oct. 2021). ISSN: 1867-2957. DOI: `10.1007/s12532-021-00208-8`. URL: `https://doi.org/10.1007/s12532-021-00208-8`.

[86] Qianhao Wang, Yuman Gao, Jialin Ji, Chao Xu, and Fei Gao. "Visibility-aware Trajectory Optimization with Application to Aerial Tracking". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5249–5256. DOI: `10.1109/IROS51168.2021.9636753`.

[87] Yue Wang, Yafeng Guo, and Jun Wang. "A hierarchical planning framework of the intersection with blind zone and uncertainty". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 687–692. DOI: `10.1109/ITSC48978.2021.9564821`.

[88] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M Rehg, and Evangelos A Theodorou. "Robust Sampling Based Model Predictive Control with Sparse Objective Information." In: *Robotics: Science and Systems*. 2018.

[89] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. "Information theoretic MPC for model-based reinforcement learning". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1714–1721. DOI: `10.1109/ICRA.2017.7989202`.

[90] Marios Xanthidis, Michail Kalaitzakis, Nare Karapetyan, James Johnson, Nikolaos Vitzilaios, Jason M. O'Kane, and Ioannis Rekleitis. "AquaVis: A Perception-Aware Autonomous Navigation Framework for Underwater Vehicles". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5410–5417. DOI: `10.1109/IROS51168.2021.9636124`.

[91] Xuesu Xiao, Bo Liu, Garrett Warnell, and Peter Stone. "Motion planning and control for mobile robot navigation using machine learning: a survey". In: *Autonomous Robots* 46.5 (2022), pp. 569–597.

[92] Xuesu Xiao, Zizhao Wang, Zifan Xu, Bo Liu, Garrett Warnell, Gauraang Dhamankar, Anirudh Nair, and Peter Stone. "Appl: Adaptive planner parameter learning". In: *Robotics and Autonomous Systems* 154 (2022), p. 104132.

[93] Zifan Xu, Xuesu Xiao, Garrett Warnell, Anirudh Nair, and Peter Stone. "Machine learning methods for local motion planning: A study of end-to-end vs. parameter learning". In: *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2021, pp. 217–222.

[94] Brian Yamauchi. “A frontier-based approach for autonomous exploration”. In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97.’Towards New Computational Principles for Robotics and Automation’*. IEEE. 1997, pp. 146–151.

[95] Pengzhi Yang, Yuhan Liu, Shumon Koga, Arash Asgharivaskasi, and Nikolay Atanasov. “Learning Continuous Control Policies for Information-Theoretic Active Perception”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 2098–2104. DOI: 10.1109/ICRA48891.2023.10160455.

[96] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. “Occlusion-Aware Risk Assessment for Autonomous Driving in Urban Environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2235–2241. DOI: 10.1109/LRA.2019.2900453.

[97] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. “Risk Assessment and Planning with Bidirectional Reachability for Autonomous Driving”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5363–5369. DOI: 10.1109/ICRA40945.2020.9197491.