Proactive and Attentive Autonomous Navigation and Avoidance of Dynamic and Intermittently-Visible Actors

A Thesis

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science (Computer Science)

by

Garrett A. Moore Jr.

December 2023

Approval Sheet

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

Master of Science (Computer Science)

Garrett A. Moore Jr.

This thesis has been read and approved by the Examining Committee:

Nicola Bezzo, Advisor

Madhur Behl, Committee Chair

Tariq Iqbal

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, Dean, School of Engineering and Applied Science

December 2023

Abstract

In the increasingly populated and dynamic world we inhabit, one of the fundamental challenges autonomous mobile robots face is navigating through crowded environments. In addition to creating a more complex environment for robots to traverse, crowds also introduce the challenge of intermittent occlusions - actors in the environment may become temporarily occluded from each others' view by other actors as they move through the environment. Intermittent occlusions can result in scenarios where actors seemingly appear out of nowhere, which can induce erratic or unsafe behavior in a robot's planned trajectory. To mitigate this risk, we propose a novel framework for identifying actors of interest in crowded environments based on observed actor dynamics and predicting their behavior over a short time horizon. These predictions are used to formulate constraints for a model predictive controller, which allows our system to compute an optimal trajectory through crowded environments containing intermittently occluded actors.

Acknowledgments

First, I would like to thank my advisor Nicola Bezzo for his support, guidance, and inspiration during my time at the University of Virginia. He has been instrumental in my academic and professional development, and has always encouraged me to explore and engage with the interests that brought me back to Graduate School in the first place. I am very thankful to have had him as my advisor and mentor.

I would also like to thank my friends and colleagues in the AMR lab - Jacob, Nick, Lauren, Patrick, Shijie, Rahul, and Will. This brilliant, kind, and hilarious group has been so supportive of me as a student and researcher and has really helped make UVA feel like a home to me over the past few years.

Finally, I would like to thank my family for their continual love and support throughout my life. They have always been there for me no matter what, and have continually encouraged me to be the best version of myself that I can be. Mom, Dad, Allison, Isaiah, and Chloë Ester, I am so thankful for you and everything you've done to support me on this journey!

Contents

List of Figures

Chapter 1

Introduction

Autonomous mobile robots are developing increasingly complex capabilities as the state-of-the-art advances. As these systems' capabilities advance, so too does the complexity of the environment in which they can be successfully deployed. One particularly challenging component of autonomous navigation through an environment is coexisting with other intelligent entities and taking their actions into consideration while carrying out an independent mission. Because many of the domains at the forefront of robotics research are densely populated by vehicles, pedestrians, and other mobile entities, this fusion of path planning and obstacle avoidance becomes increasingly complex as more and more actors enter the environment.

While many methodologies exist for motion planning around dynamic actors, real-world environments have an additional facet that further complicates the problem – imperfect information. The sensors most typically deployed on autonomous mobile robots, Lidar, Radar, and Cameras, can only sense objects in their line of sight. In densely populated environments like those mentioned above, actors are often occluded by terrain, obstacles, and even other actors. These occlusions are often intermittent and difficult to predict, such as pedestrians crossing through traffic, large buses or trailers passing smaller vehicles on the highway, or off-road robots passing through dense foliage in the woods. An example of these sorts of occlusions is shown in Figure 1.1. Traditional motion planning algorithms only consider the actions of visible actors, so in occlusion-rich environments, autonomous systems risk failing to consider the behavior of actors outside of their current field of view (FOV). Actors disappearing and reappearing at unexpected times can lead to scenarios in which motion planners are forced to react at the last minute, which can result in drastic or erratic behavior. This is a significant shortcoming that can pose a substantial risk to robots and the other entities with which they share their environments.

When we as humans navigate through crowded environments, we can implicitly separate the signal from

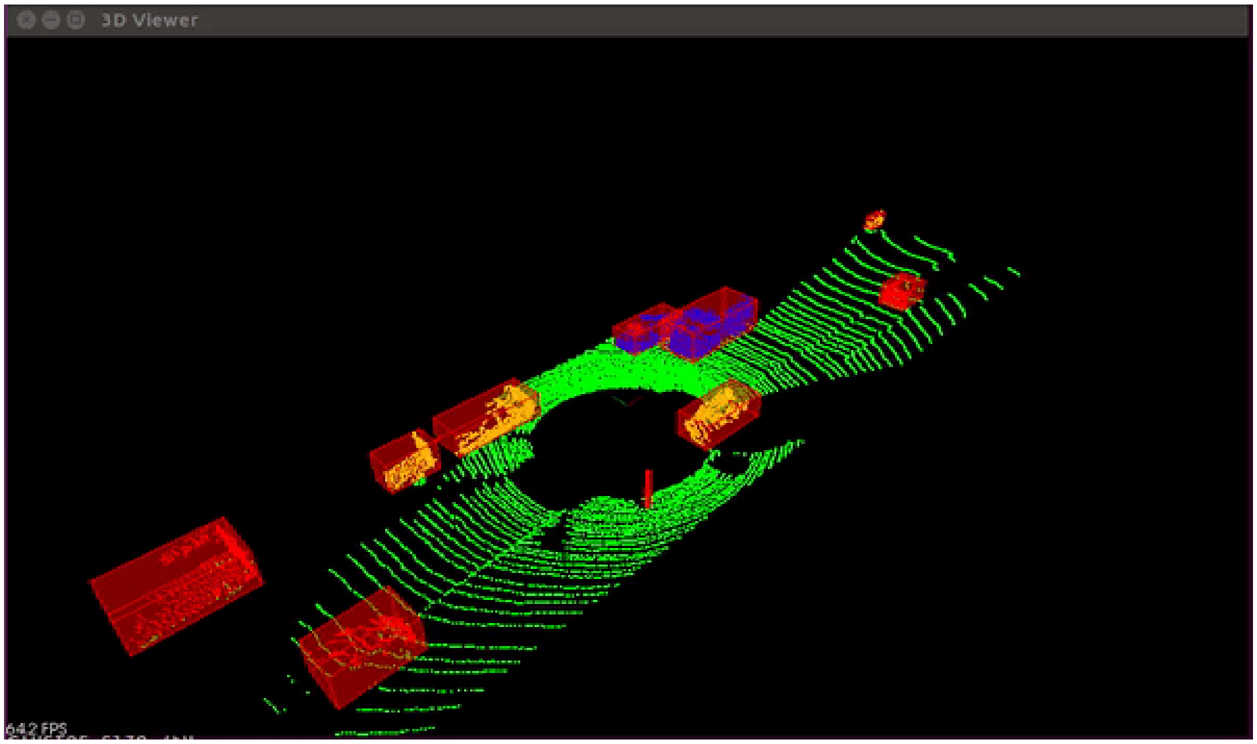


Figure 1.1: LiDAR field of view becoming intermittently occluded by highway drivers[5]

the noise; through our observations, we can determine which elements of our environment we should pay attention to and which elements we can safely ignore. This process is known as Saliency Detection [26] and is a crucial process for efficiently negotiating crowded decision spaces. Notably, our ability to reason about the actions of entities in our surroundings is not limited to what we can currently see. Based on our previous observations, we are able to infer what may be happening within regions of our surroundings that are occluded or otherwise hidden from us. There are many different ways humans accomplish this, but one specific way that we will examine in this thesis is simple dynamic prediction. If we observe an actor moving out of sight, we begin to predict the trajectory that that actor could follow based on our observations about that actor and our knowledge of likely objectives in the surrounding environment. In simple scenarios, this could be as straightforward as projecting along the actors' last observed trajectory at its last observed velocity; in more complex cases, this could be projecting along multiple possible trajectories based on our assessment of viable goals in the environment. The common thread is that our decisions are informed by predictions about actors' future intentions given observations about their dynamics.

Inspired by this, we present a novel motion planning approach that is able to determine whether or not an actor is relevant to our path planning computations. If an actor that is determined to be relevant becomes occluded from our view, our approach utilizes the actor's last observed dynamics to predict its most aggressive possible trajectory out to a finite time horizon. The positions in this trajectory are used as constraints for a

Model Predictive Controller (MPC), which allows for proactive navigation around states where the occluded actor(s) would be if they followed their most aggressive trajectory.

This thesis is organized in the following manner: Chapter 1 presents an introduction to the thesis and an overview of its core contributions. Chapter 2 reviews the state-of-the-art for dynamic obstacle avoidance and occlusion-aware obstacle avoidance. Chapter 3 discusses the problem preliminaries and mathematical formulation, and Chapter 4 discusses the specifics of our proposed approach in depth. Chapter 5 gives an overview of the simulated case studies, real-world experiments, and their results. Finally, Chapter 6 provides a conclusion to the thesis, along with a discussion of potential future lines of effort to advance this research.

1.1 Contribution

The two primary contributions of this thesis are as follows:

Occlusion-Aware Trajectory Prediction for Dynamic Actors — The first contribution of this work is a motion planning framework that utilizes observations about self and actor dynamics to avoid collisions while traveling to a goal pose. This framework is a novel extension of a Model Predictive Controller (MPC) that recognizes when salient actors have become occluded from the robot's field of view. Once an occlusion has been detected, our approach uses the last observed state of the occluded actor to predict the worst-case trajectory that it could follow over a finite time horizon - the trajectory that would minimize the robot's reaction time when the actor reappears in the robot's field of view. This predicted trajectory consists of a series of actor positions over the MPC's prediction horizon, which are then used to form constraints in the underlying MPC. In this way, the MPC is able to safely plan a path around each actor's predicted locations at each step in its prediction horizon.

Attentive Actor Tracking — While the previous contribution serves as a powerful tool to plan safe paths around predicted actor positions, it suffers from an increase in complexity proportional to the number of actors that the motion planner is tracking. In order to combat this, we have developed an attentive component to our dynamic predictions that allows the planner to disregard actors that will not likely be relevant to the robot's planned trajectory. This is accomplished by monitoring the relative motion of actors in the robot's environment. All visible actors are initially treated as 'tracked' by the motion planner, but if the distance between an actor and the robot is observed to be continuously increasing over a set time interval, the planner stops tracking that actor. Only the trajectories for tracked actors are predicted in the Dynamic Prediction module, and subsequently used as constraints for the underlying MPC. This ensures that resources are not spent computing trajectories for actors that are unlikely to interact with the robot over its control horizon.

Chapter 2

Survey of Related Work and State of the Art in Dynamic Obstacle Avoidance

Motion planning and collision avoidance are highly studied topics in the field of autonomous mobile robotics and are fundamental to the operation of autonomous systems. Methods for avoiding dynamic obstacles are particularly important to systems that operate in human-inhabited spaces, as humans are highly mobile agents with a wide variety of potential behaviors. In this section, we will discuss the current state-of-the-art practices for obstacle-avoidant motion planning, along with some of their key strengths and deficiencies. We will also discuss current state-of-the-art practices for attentive obstacle tracking, and methods by which complex scenarios with many agents can be reduced to simpler terms.

2.1 Path Planning for Dynamic Obstacle Avoidance

There are many popular approaches to path planning that are utilized for obstacle avoidance in robotics today, ranging from force-based methods like Artificial Potential Fields [27] and Time Elastic Bands [13], to sample based methods like RRT[17] and PRM[9], to various forms of optimization-based planning such as Reinforcement Learning[19], Control Barrier Functions [23], and Model Predictive Control[7].

While many of these optimization-based planning methods have previously been relegated to offline usage in the past due to their computational expense, the ever-expanding capabilities of embedded computing platforms have meant that many of these methods are now beginning to become accessible solutions for online, real-time path planning.

Chief among the strengths of optimization-based planners is their ability to optimize around diverse sets of constraints, a quality which makes them well-suited for solving path planning problems in complex environments. Incorporating obstacle state information as a constraint is common practice in many obstacle avoidance approaches, but most of these approaches only take into account the states of obstacles currently visible to them - in this case, within the range of their sensors.

This poses a significant limitation to these path planning methods when operating in environments with imperfect information, particularly in circumstances where obstacles are intermittently visible. Our approach aims to address this deficiency by introducing a form of occlusion handling to predict the trajectory of intermittently occluded actors in the environment.

2.2 Occlusion Handling

Occlusion handling is a specialized branch of motion planning that accounts for the uncertainty inherent to areas that are outside of a robot's field of view (FOV). Many approaches such as [6] utilize Partially Observable Markov Decision Processes (POMDP), which allows them to plan an optimal path over a distribution of probable states of occluded actors, but the complexity scales linearly with the number of actors in the environment. Authors of [4] improve on this approach by independently optimizing around each actor and then fusing the results, but their approach is reliant on knowing the number and type of actors in the environment a priori. Authors of [1] consider robot and environment dynamics to compute worst-case velocity profiles for occluded actors, but their approach was only ever able to be implemented as an offline planner. Authors of [11] developed an optimization constraint that allows an MPC to minimize the occluded area as it navigates by occlusions, both reducing the risk of collision and supporting faster motion around occluded obstacles.

Most of the current approaches for handling safe navigation around occlusions work well with fewer occlusions but have difficulty scaling to environments with larger numbers of potential occlusions and occluded actors. Our approach seeks to reduce the severity of this difficulty by intelligently reducing the number of actors that we need to predict motion for.

2.3 Attentive Obstacle Avoidance

Attentive obstacle avoidance is another topic that has been explored in recent years in an attempt to determine which sensed entities are most significant or salient in a given environment. Authors in [24] aim to emulate the human capability to identify the most salient objects in a scene in a LiDAR-based perception framework, and authors in [10] seek to identify scale-able salience perception algorithms. Other frameworks such as SARL [18] utilize deep neural networks to perform socially attentive pooling, which aims to identify the salience of each individual agent in a crowded environment. Authors of [25] take a broader approach, applying principles of group-based behavior prediction to construct an MPC that plans around observed group dynamics. [20] expands on this effort, coupling group classification with a robust MPC that also accounts for actor dynamics. While many methods in this field that utilize machine learning to develop classification tools for determining salience are highly robust, one downside is that they often require significant amounts of training to learn specific scenarios. In our approach, we explore a simplified concept of salience that does not require training machine learning models, and instead analyzes trends in actors' dynamic behavior over time. Our goal is to develop a salience determination process that is general enough to support the wide variety of actors that an autonomous mobile robot might encounter while navigating in a crowded urban environment.

2.4 MPC-Based Motion Planning

We found MPC to be an appealing methodology to explore for our application due to its ability to perform online optimizations around multiple constraints while taking into account the dynamics of the robot and the actors around it. During our research, we encountered many cases where MPC was utilized in support of objectives within the domains we were already researching.

Authors of [16] utilize an MPC-based solution to track an obstacle avoidance trajectory generated by a potential field planner. [12] implements an MPC-based planner to plan a path that minimizes the area of occluded regions and predicts the probability of colliding with actors appearing from the occluded regions. The author of [20] provides inspiration for utilizing MPCs in an attention-aware motion planning context, using a deep neural network based approach to reduce the set of actors that an MPC needs to predict motion for.

While MPC-based controllers can serve as excellent motion planning solutions in many scenarios, they are not without their challenges. The computational complexity of their optimizations can grow rapidly with the number of constraints they must optimize around and the time horizon they must optimize over. Because the optimization is solved online, increased latency poses the risk of introducing latency into a robot's control system. Moreover, many MPC-based obstacle avoidance methods only account for obstacles that are visible to the robot. In crowded and occlusion-rich environments, this does not remain sufficient for safe motion planning. Obstacles not visible to the robot are often excluded from the optimization process. Our approach aims to address these deficiencies by combining occlusion handling and attentive obstacle avoidance methodologies to produce an MPC-based controller that can both determine salient obstacles within a robot's surroundings and predict their trajectories when they become occluded from the robot's FOV.

Chapter 3

Problem Formulation

This chapter provides an overview of the problem space explored by our work. We will begin with a discussion of the notation and formulas used to describe the problem and our systematic approach, followed by an overview of the fundamental techniques expanded upon by this thesis. This will be followed by a formal problem definition that describes the challenge that this work aims to address.

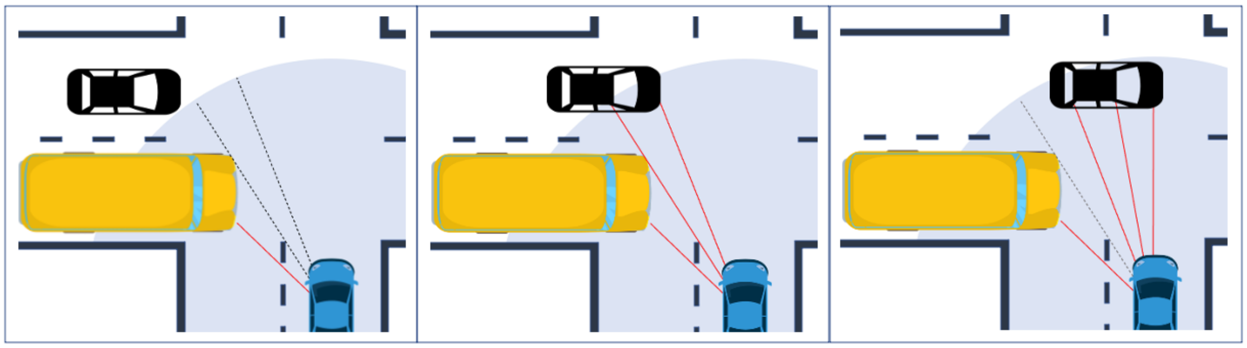


Figure 3.1: The black car is temporarily occluded from the blue car's view by the yellow bus

3.1 Problem Formulation

Consider an autonomous robot navigating through a structured or unstructured environment populated by other entities referred to as actors. This environment contains both static and dynamic actors that are carrying out tasks independent of that of the robot, and due to the nature of the environment these actors can become intermittently occluded from the robot's field of view. An example of intermittent occlusion is illustrated in Figure 3.1. These actors are assumed to be non-cooperative and non-communicative with the robot. We are tasked with developing a motion planning and control policy to navigate to a goal pose while minimizing the risk of collision with relevant actors in our vicinity.

Due to the unpredictable intermittent visibility of the actors in the environment, we are faced with a challenging motion-planning problem. When an actor suddenly appears out of nowhere within close proximity to the robot, the resulting motion planning computations can produce sudden and erratic control policies. In the real world, this chaotic behavior can be dangerous to both the robot and the actors around it, resulting in, at best, an uncomfortable experience and, at worst, a collision incident.

Furthermore, as the number of actors in a scenario grows, so too does the computational complexity of the optimal control problem. If the number of actors in the robot's vicinity grows to be too large, the robot may become unable to carry out safe motion planning efficiently enough to operate at near real time.

The problem, then, becomes the following: how can we develop a motion planning and control policy to navigate to a goal while minimizing the risk of collision with significant actors in our vicinity, and how can we intelligently reduce the number of actors that must be considered in our control optimization?

Our formulation of this problem is based on the following assumptions:

1. The Robot's perception is limited to a generalized line-of-sight model that represents a fusion of lidar and camera data.

2. The Robot's perception capabilities are able to accurately measure actor position, orientation, and velocity

3. The Robot possesses a priori knowledge of the environment's structure via a map and is able to reason about structured trajectories that the actors can take.

4. The Robot does not explicitly communicate with the actors

Chapter 4

Approach

In this chapter, we describe our technical approach to motion planning through an environment populated by dynamic and intermittently occluded actors. Our framework consists of two core components: First, our attentive actor tracking module uses observations of actors' dynamics to determine which actors are considered salient to the robot's motion planning process. These actors are marked as 'tracked'. Next, the set of tracked actors is passed to the dynamic prediction module, which uses the tracked actors' last observed dynamics to predict their future trajectories over a short time horizon. If a tracked actor becomes occluded from the robot's field of view, the prediction module performs an additional layer of prediction, in which a sample of viable trajectories is used to generate an approximate reachable set of positions for the occluded actor. The trajectories and regions generated in the dynamic prediction module are used as constraints in a model predictive controller, which is then able to compute an optimal control policy over a fixed time horizon that avoids all tracked actors in the robot's vicinity.

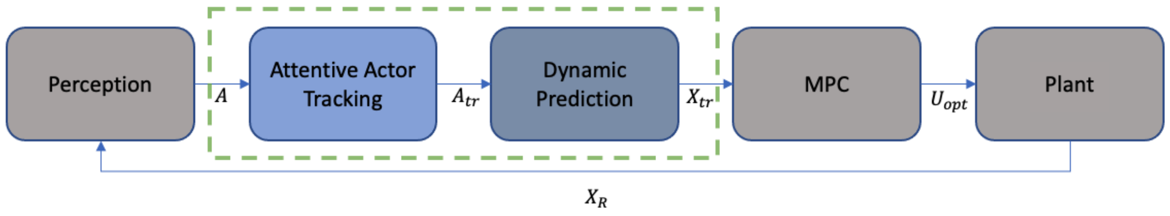


Figure 4.1: System block diagram, with core contributions outlined in green

As indicated in Figure 4.1 above, the perception module senses the robot's environment and returns a set of the positions and velocities of all actors within the range of the robot's sensor package, A. The Attentive Actor Tracking module determines which members of A should be tracked and which previously tracked members should be untracked. This module returns A_{tr}, the set of last known positions and velocities of all tracked actors at time t. The Dynamic Prediction Module uses A_{tr} to predict a set of future positions for each tracked actor, X_{tr}, which are then passed to the MPC to be formulated into obstacle avoidance constraints. After updating its constraints, the MPC computes an optimal control sequence for the robot to reach its goal and issues those controls to the robot (plant). Finally, the plant executes the commands in the real or simulated world and publishes its updated state X_R back to the Perception module. In order to illustrate the approach discussed in this chapter, we will provide images from our Matlab simulations. These simulations are included as a reference in this section but will be discussed in greater depth in Chapter 5.

4.1 Perception

The perception process controls how the robot receives information about its surroundings, and what information it can use to perform the rest of the computations in our approach. This section discusses the data produced by the perception process, and how it is passed to subsequent modules in the approach.

4.1.1 Actor State Perception

First, the robot measures the state of each actor in its environment. The robot assumes that it receives observations about each actor a within the robot's perception range that contain that actor's position $[x^a, y^a]$ and velocity $[v_x^a, v_y^a]$. Independent of perception, the robot also maintains a Boolean Φ_{tr}^a that describes whether or not actor a is tracked by the Attentive Actor Tracking module. The concept of tracking will be explained in detail in the following section. Each observation is stored as a state matrix X_a, where $X_a = [x^a, y^a, v_x^a, v_y^a, \Phi_{tr}^a]^T$. Lastly, each of these matrices is added to an overall state matrix X that contains the collective states of all visible or tracked actors.

$$\mathbf{X} = \begin{bmatrix} \begin{bmatrix} x^1 \\ y^1 \\ v_x^1 \\ v_y^1 \\ \Phi_{tr}^1 \end{bmatrix} & \cdots & \begin{bmatrix} x^n \\ y^n \\ v_x^n \\ v_y^n \\ \Phi_{tr}^n \end{bmatrix} \end{bmatrix}$$

4.1.2 Actor Visibility

Since the robot relies on a line-of-sight sensor for perception, we anticipate that there will be intermittent visibility of actors in the robot's environment. The robot maintains a list of visible actors in its environment at each cycle, along with the time that they were last visible, which allows it to maintain a record of any intermittent occlusions occurring during operation. The following Figure 4.2 illustrates how the robot visualizes its environment based on the data it receives from its perception module. In this figure, the robot is represented by the blue triangle inside the red circle, and each visible actor is represented by a black circle. The white region represents the visible area within the robot's perception range. Note that the visible area is limited by both the sensor's range and physical interference from the sensed actors.

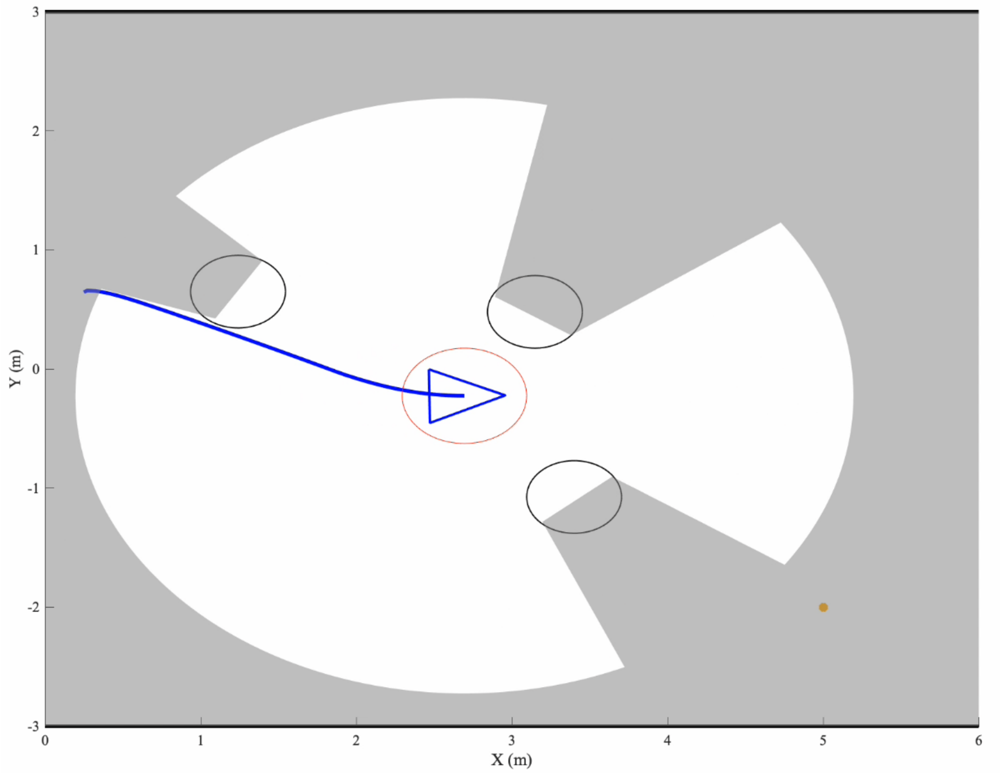


Figure 4.2: Sample perception data from simulated line-of-sight sensor

4.1.3 Last Known State

Combining typical state measurements with visibility awareness allows the robot to maintain a set of *Last Known States* for each actor that they've encountered in their environment, X_L. This contains the last observed position $[x^a, y^a]$ and velocity $[v^a_x, v^a_y]$ of each actor, its corresponding tracking variable Φ^a_{tr}, and the time it was last observed at t. The Last Observed State, X_L, is represented as follows:

$$\mathbf{X_L} = \begin{bmatrix} \begin{bmatrix} x^1 \\ y^1 \\ v_x^1 \\ v_y^1 \\ \Phi_{tr}^1 \\ t^1 \end{bmatrix} & \dots & \begin{bmatrix} x^n \\ y^n \\ v_x^n \\ v_y^n \\ \Phi_{tr}^n \\ t^n \end{bmatrix} \end{bmatrix}$$

This contains all the information necessary for the robot to perform this approach. In the following section, we will discuss how this is utilized to determine which actors the robot should track, make predictions about future actor positions, and maintain those predictions throughout intermittent occlusions.

4.2 Dynamic Prediction of Actor Trajectories

The foundational element of our approach is dynamic path prediction. As the robot observes its surroundings, it constantly measures the positions of all actors within its sensing radius. The positions of the observed actors over time are used to compute velocity estimates for each actor, and these velocities are subsequently used to predict a series of reachable future states for each actor over a receding time horizon.

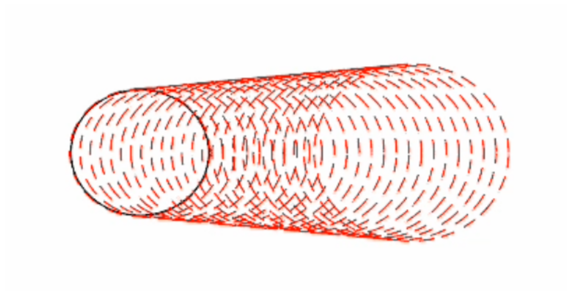


Figure 4.3: Basic dynamic prediction for an actor

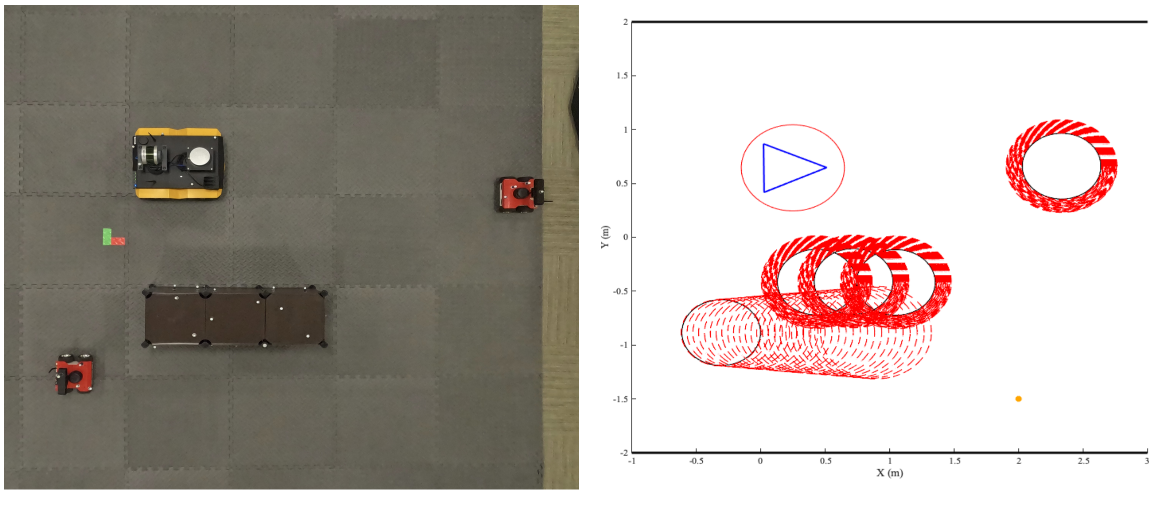
Each actor a in our environment is assigned a safety radius r^a that the MPC must respect as it plans its path, which is defined from the dimensions of the actor measured by the robot's perception module. As seen in Figure 4.3 above, the robot represents its future state predictions as a series of circles around each actor's reachable future states, one for each step in the MPC's prediction horizon. The positions of these circles' origins are predicted using the kinematic equations below. At each prediction step, the radius of the circle increases by a scale factor f_r to account for uncertainty in the future state of the actor. This margin accounts for uncertainties in the direction of motion, process error in executing the optimal control commands, and noise due to disturbances encountered by the actor as it moves along its trajectory. This scale factor f_r serves as an approximation for computing the reachable set of states given that combined uncertainty. This is not expected to be a permanent replacement for a more formal reachability analysis tool but rather serves as a useful approximation that can be tuned via experimentation.

The resulting prediction is presented as a sequence of circles with gradually increasing radii, as seen in Figure 4.3 above. This set of actor positions and radii is added to X_{tr}, which will allow the MPC to plan a trajectory that respects the boundaries of each actor at each step in its control horizon.

$$x^{a\prime} = x^a + v_x^a dt \tag{4.1}$$

$$y^{a\prime} = y^a + v_y^a dt \tag{4.2}$$

By default, we perform this prediction for each actor in our environment within our line of sight. Figure 4.4 shows a series of predictions for an environment containing three entities: two robots and a large wall structure. The ego vehicle is represented by a blue triangle inside a circle, and each actor is represented by a black circle. In order to better approximate a safe boundary around larger objects like the wall shown in this example, we discretize them into smaller objects and treat each component as an independent actor. As shown in Figure 4.3, each actor's dynamic prediction is represented by a series of red circles starting at each actor's origin and extending along the actor's observed velocity vector.



(a) (b)

Figure 4.4: Real-world experiment setup (a) next to a visualization of the robot's dynamic predictions for nearby actors (b)

While this approach is adequate on its own for small numbers of actors, it quickly grows in computational expense and complexity with the number of actors being tracked. The following section describes our approach to reducing this complexity using Attentive Actor Tracking.

4.3 Attentive Actor Tracking

The Attentive Actor Tracking module determines which actors in the robot's environment are worth paying attention to and which actors are not. This is significant because being able to ignore actors reduces the number of actor trajectories that need to be predicted by the Dynamic Prediction Module, and also significantly reduces the number of constraints passed to the MPC.

We begin by assuming that the robot should always pay attention to visible actors that are moving consistently towards the robot's position and can always disregard visible actors that are moving consistently away from the robot's position. This can be achieved by continuously measuring the Euclidean Distance, d_e^i, (eq. 4.3) between each visible actor i and the robot, where the robot's position is represented by $[x_r, y_r]$ and the position of each actor is represented by $[x_a^i, y_a^i]$.

$$d_e^i = \sqrt{(x_r - x_a^i)^2 + (y_r - y_a^i)^2} \tag{4.3}$$

$$\Phi_{tr}^i = \begin{cases} 0 & d_e^i(t) > d_e^i(t-1)\ \forall t_0 < t < n_t \\ 1 & \text{otherwise} \end{cases} \tag{4.4}$$

Recall from Section 4.1.3 that Φ_{tr}^i is a boolean that represents whether or not actor i is tracked by the robot. Equation 4.4 serves as a way to continuously recompute that value at runtime.

This computation is performed at each cycle of the robot's control process. If the computed distance to an actor has increased over each of the previous n_t cycles, then the robot determines that it no longer needs to pay attention to that actor. The variable n_t can be adjusted experimentally to tune the performance of the Attentive Actor Tracking Module but is set to 5 cycles by default.

In order for our approach to be aware of whether or not the robot is paying attention to an actor, the robot must maintain a tracking variable Φ_{tr}^i for each actor i. In our framework, an actor is considered to be *tracked* ($\Phi_{tr}^i = 1$) immediately upon entering the robot's field of view. At each cycle the robot recomputes d_e^i between the robot's position and each actor i. If d_e^i is increasing and has been continuously increasing for the past n_t cycles, then actor i becomes *untracked* ($\Phi_{tr}^i = 0$). Similarly, if d_e^i between our position and untracked actor i is decreasing and has been continuously decreasing for the past n_t cycles, then actor i becomes *tracked* again.

This process can be visualized in the following scenario, shown in Figure 4.5. As the robot navigates through the environment toward its goal (indicated by the orange point), it initially tracks all five actors and performs dynamic trajectory prediction for each of them. For the tracked actors, this future set of positions is represented by a sequence of red circles. As we pass the leftmost actor i, d_e^i begins increasing, and after n_t cycles we untrack actor i.

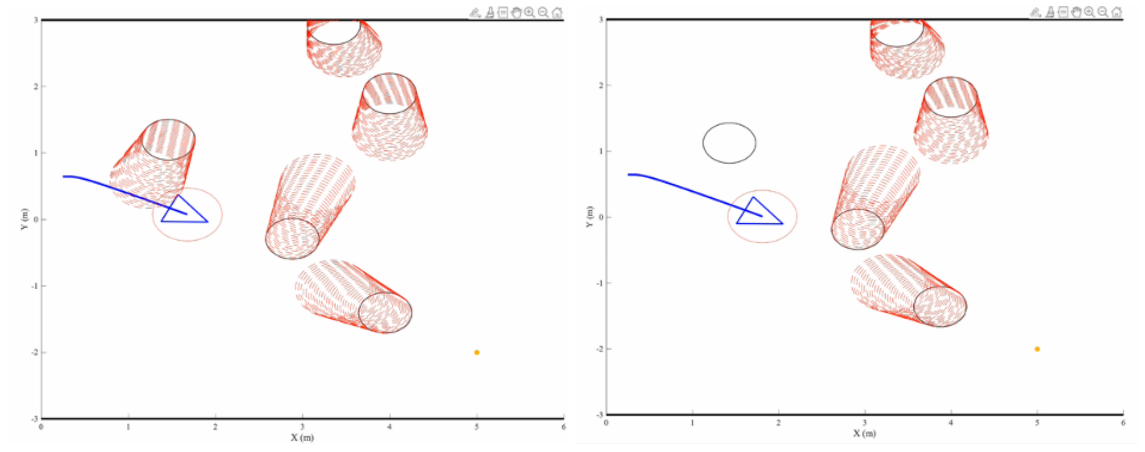

(a) (b)

Figure 4.5: Sample scenario of tracking five actors (a) then untracking the leftmost actor (b).

4.4 Occlusion-Aware Dynamic Prediction Module

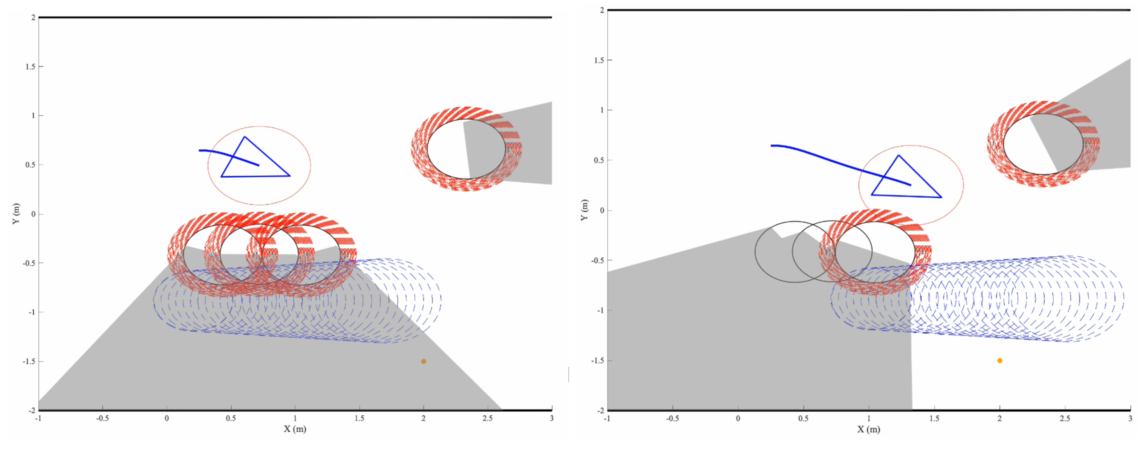
This section describes the function of the Occlusion-Aware Dynamic Prediction Module, which predicts a sequence of future positions X_{tr} for each tracked actor in the set A_{tr}. This builds on the basic Dynamic Prediction process described in section 4.2 but also incorporates the Last Observed State of each actor to predict the trajectory of occluded actors.

4.4.1 Dynamic Prediction for Occluded Obstacles

Dynamic predictions for occluded obstacles are accomplished in a similar manner to predictions for visible obstacles. The primary difference is that since the robot can no longer truly observe the state of the tracked actor when it is occluded, its predictions must also account for changes in velocity or trajectory over the duration of the actor's occlusion.

Our approach begins by estimating an upper bound for a velocity that the occluded actor could reasonably achieve, V_{ub}. This value is experimentally determined to be an appropriate maximum for the desired use case.

Once a tracked actor becomes occluded, the Dynamic Prediction module predicts future states of the occluded actor over the MPC's control horizon using V_{ub}, starting from the last observed state of the occluded actor i, X_L^i. This predicted state of the occluded actor n is propagated from cycle to cycle using V_{ub} and the actor's equations of motion, but if at any time the predicted origin of the occluded obstacle enters the observable area of the robot, this indicates that the prediction was invalid. In this case, V_{ub} is reduced by a constant percentage δ_v, and the predicted path is recomputed from the last observed state of the robot. δ_v can be tuned experimentally to adjust performance. In this way, the predicted position of the occluded actor converges to just outside the visible area of the ego vehicle. As the visible area of the robot's perception module continues to recede, the fastest predicted state of the occluded actor converges with the robot's actual position.



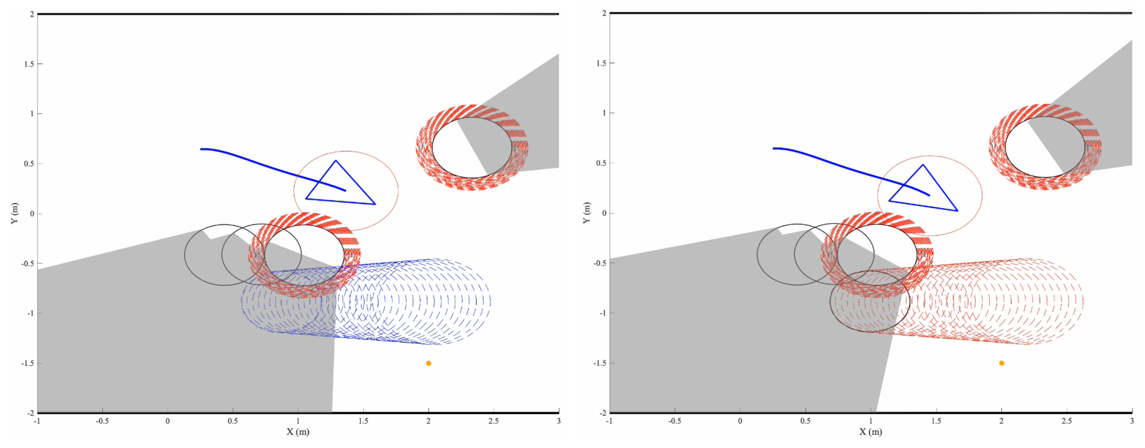
(a) (b)

Figure 4.6: The dynamic prediction for the occluded actor (a) enters the robot's perception range (b).

Consider the continuation of the scenario shown in Figure 4.4. As the mobile actor i becomes occluded (shown in Figure 4.6(a)) the robot computes a Dynamic Prediction for actor i with increased velocity V_{ub}. This Dynamic Prediction continues at each time step until the moment where the initial position of that prediction enters the visible area within our perception radius (shown in Figure 4.6(b)). If this predicted trajectory had been accurate, the robot would have sensed the occluded actor reappear at this point, so this indicates that we need to recompute our prediction and propagation with a lower V_{ub}.

We update our velocity upper bound such that V_{ub} = (1 - δ_v)V_{ub} and recursively recompute our predictions from the last known position of actor i until the initial position of the prediction is no longer visible. This position is then propagated until the initial position in this prediction becomes visible, and in this case, we observe the predicted trajectory converge with the actual trajectory of actor i as it reappears.

In the resulting prediction, shown in Figure 4.7, the actor's predicted position converges to a position just outside the robot's field of view inside the occluded region. This results in an approximation of the worst-case scenario for that actor's trajectory, the trajectory where the occluded actor will reappear out of the occluded region at the next time step. We consider this the worst-case scenario because it minimizes the



(a)

(b)

Figure 4.7: Prediction is recomputed (a), then converges with the occluded actor's actual state (b).

distance and time that the robot has to react to the actor when it reappears, which increases the likelihood that the robot's motion planner will generate an erratic path in response to a sudden constraint.

The dynamic prediction module produces a set of future positions for each tracked actor, X_{tr}. This set contains the series of predicted states for each tracked actor at each time step t from $1 < t < N$, where N is the control horizon we're using for our MPC. This will allow us to construct constraints for our MPC that represent the state of each actor at each time step, an operation that is critical for avoiding dynamic obstacles. A deeper discussion of how this is utilized and a further discussion of the overall MPC will be the focus of the next section.

4.5 MPC-Based Obstacle Avoidance

The Model Predictive Controller (MPC) is a predictive control approach that computes a set of optimal control inputs that minimize a particular cost function. This cost function consists of formalized mathematical constraints, which can be tailored to describe the requirements of a given application. For our application, we have constructed an MPC capable of avoiding dynamic obstacles. As a part of this process, the controller makes predictions about future trajectories of both the robot and its surrounding actors using dynamic models then utilizes those predicted trajectories as time-variant position constraints in its control optimization process. The MPC's optimization constraints can also consider operational elements such as safety margins around actors, simple obstacle avoidance behaviors, lane lines, and proximity to a goal pose. The output of the Dynamic Prediction Module, X_{tr}, is passed into our Model Predictive Controller (MPC) as a set of position predictions over a time horizon, N. This set of predictions is used to construct position constraints, which allows us to optimize around them over the duration of the time horizon.

Kinematic Models

One of the foundational components in the formulation of our Model Predictive Controller (MPC) is the set of kinematic models that describe the motion of the robot and the actors that it interacts with. The robot's motion is modeled with a non-holonomic unicycle model [8]. Using the unicycle model, we represent the state of the robot as follows: $\mathbf{x_r} = [x_r, y_r, \theta_r]^\intercal$, where x_r refers to its x-axis position, y_r refers to its y-axis position, and θ_r refers to its orientation. The robot's control input is represented as follows: $\mathbf{u_r} = [v_r, \omega_r]^\intercal$, where v_r represents a linear velocity and ω_r represents an angular velocity. Thus, the robot's equations of motion are as follows:

$$\begin{aligned} \dot{x}_r &= v_r \cos\theta_r \\ \dot{y}_r &= v_r \sin\theta_r \\ \dot{\theta}_r &= \omega_r \end{aligned} \tag{4.5}$$

Next, we define the kinematic model for all obstacles using a point mass kinematic model. The state of each actor is represented as $\mathbf{x^a} = [x^a, y^a]$, and the control input issued to each actor is represented as $\mathbf{u^a} = [v_x^a, v_y^a]$, where each term represents a velocity command issued in the x and y directions respectively.

Thus, the equations of motion for the actors are as follows:

$$\begin{aligned} \dot{x}^a &= v_x^a \\ \dot{y}^a &= v_y^a \end{aligned} \tag{4.6}$$

This simplistic model was chosen because it represents the observation that the robot is able to make about actors in its vicinity; one of our grounding assumptions is that our framework makes observations about its surroundings in the form of position and velocity. Our system is not able to determine any deeper insight into the actors' dynamics from its sensor array.

4.5.1 Formulation of Constraints

After receiving X_{tr} from the Dynamic Prediction Module, we are able to fully define the constraints that govern our optimization. We begin by creating a constraint that represents the robot dynamic model derived in the previous section. This constrains each subsequent robot state to be a function of the previous robot state and the previous control input.

$$x(t+k+1) = f(x(t+k), u(t+k)), \forall k = [0, N] \tag{4.7}$$

Next, we construct an obstacle avoidance constraint using $A_{tr}(t)$. This constraint explicitly states that for each actor in the set of tracked actors (both visible and invisible to the robot) at time t, $A_{tr}(t)$, the Euclidian distance between the robot and that actor must be greater than a safety margin δ_s. This margin is defined as follows:

$$\delta_s = r_r + r_a^i(1 + \frac{k\beta}{N}) + \rho, \forall k = [0, N], \forall i \in A_t(t) \tag{4.8}$$

Where r_r is the robot's safety radius, r_a^i is the safety radius of actor i, β is an inflation factor that approximates the effects of uncertainty on the motion of each actor, and ρ is a configurable factor of safety that increases the clearance that the robot gives other actors. Once we've defined δ_s, we can define our full obstacle avoidance constraint:

$$||x_r(t+k) - x_i(t+k)|| > \delta_s, \forall k = [0, N], \forall i \in A_{tr}(t) \tag{4.9}$$

Finally, we create boundary constraints that limit the range of states and controls within their respective feasible regions. These constraints take into account the limits of possible commands, as well as the boundaries of the traversable area (such as walls and non-actor obstacles).

$$u(t+k) \in U(t+k), \forall k = [0, N-1] \tag{4.10}$$

$$x(t+k) \in X(t+k), \forall k = [0, N] \tag{4.11}$$

4.5.2 Formulation of the Optimal Control Problem

Our goal is to compute a set of optimal controls to navigate through an environment containing heterogeneous dynamic actors. We've chosen to utilize a Model Predictive Controller (MPC) because it produces a control policy that takes into account the dynamics of the robot and the obstacles it's avoiding over time. To begin construction of our MPC, we begin with a typical MPC formulation. Given the constraints defined above, we can construct the following cost function:

$$J = (x_r(t+N) - x_g)^\intercal Q(x(t+N) - x_g) + \sum_{k=1}^{N-1} (x_r(t+l) - x_g)^\intercal Q(x(t+N) - x_g) + u_r(t+k-1)^\intercal R u_r(t+k-1) \tag{4.12}$$

In which $x_r(k)$ is the robot state at time k, x_g is the robot's goal state, $u_r(k)$ is the kth computed optimal control input, Q is the cost weighting matrix penalizing deviation in state, R is the cost weighting matrix penalizing deviation in control, and N is the control horizon. After defining the cost function, we are able to fully formulate our optimal control problem:

$$
\begin{aligned}
\underset{u(0),...u(N-1)}{\arg\min} \quad & J(x(0), u(0), ..., u(N-1)) \\
& \text{subj. to:} \\
& x(t+k+1) = f(x(t+k), u(t+k)), \forall k = [0, N] \\
& ||x_r(t+k) - x_i(t+k)|| > \delta_s(t), \forall k = [0, N], \forall i \in A_{tr}(t) \\
& u(t+k) \in U(t+k), \forall k = [0, N-1] \\
& x(t+k) \in X(t+k), \forall k = [0, N]
\end{aligned}
$$

Solving this optimal control problem results in a set of optimal controls at each time step in the MPC's control horizon, U. U consists of a series of linear and angular velocity commands that, when issued to the robot, produce an optimal trajectory that advances us towards our goal while avoiding the trajectories of the visible and invisible salient dynamic actors in our environment. Revisiting the scenario mentioned in Figure 4.7(b), we can visualize that trajectory with the green dashed line shown in front of the robot in Figure 4.8 below.

In the following section, we will explore how our approach performs through a variety of simulated and real experiments.

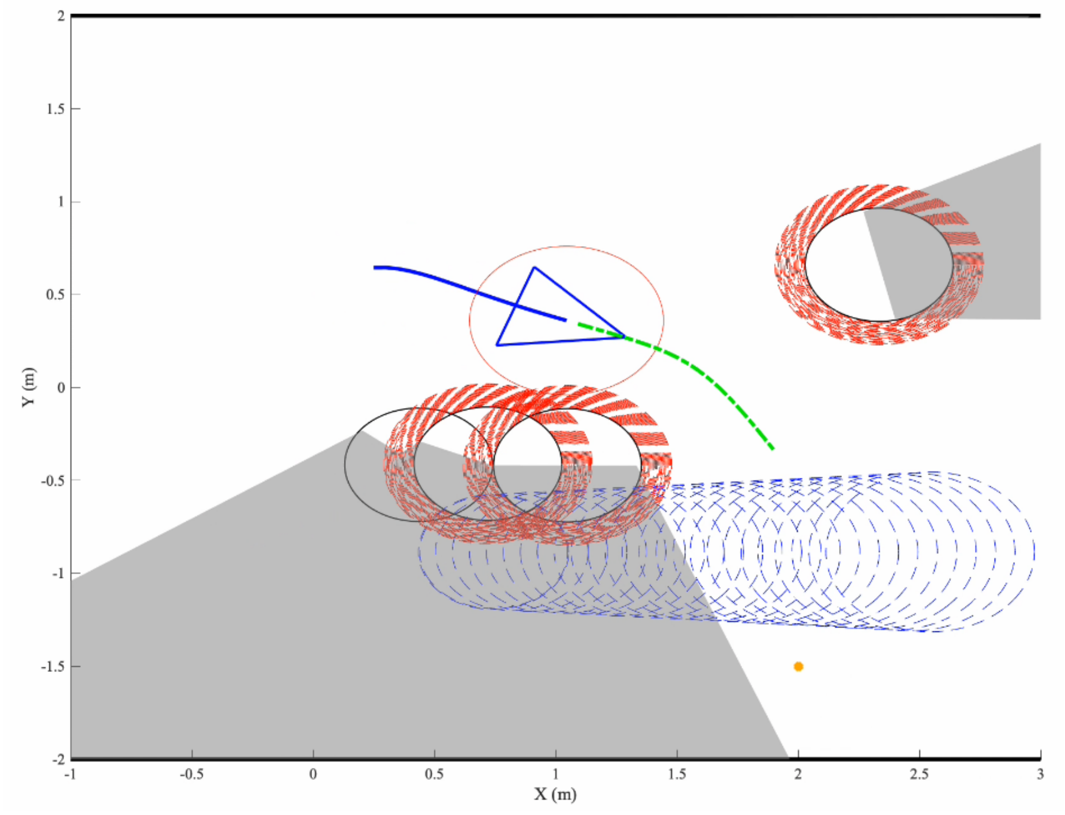


Figure 4.8: Example optimal trajectory produced by MPC

Chapter 5

Simulations and Experiments

In this section, we perform a combination of simulations and real-world experiments to validate our approach. In each of these scenarios, the robot begins in an environment containing a group of dynamic agents and is tasked with navigating around these agents to reach a goal pose. As an additional challenge, some number of the actors in each case experience intermittent occlusion from the ego vehicle's view. We begin with an overview of our software architecture and experimental setup, and then proceed to the discussion of the individual experiments and their results.

5.1 Software Architecture

The software architecture for our approach is shown in Figure 5.1 below. An in-depth description of each of these components' functions and what they're responsible for is covered in Chapter 4, but this description will serve as a higher-level overview of how these components are implemented and connected together.

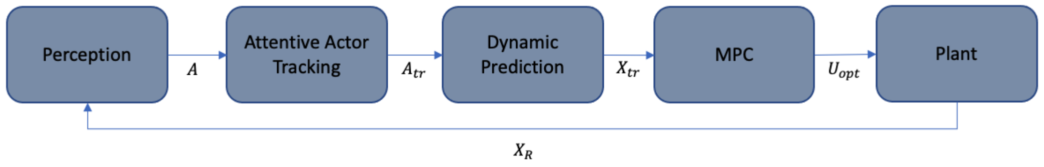


Figure 5.1: Software Architecture

The first four components in the diagram, Perception, Attentive Actor Tracking, Dynamic Prediction, and the MPC, were all implemented as part of a MATLAB R2022B [15] software package that we developed throughout the course of this research. MATLAB was chosen due to its ease of use and the robustness of its visualization capabilities, which were both helpful qualities during the development of our approach. We implemented the symbolic solver at the core of our MPC in CasADi [2], an open-source nonlinear optimization toolkit. For our simulated experiments, we simulated our plant and world within MATLAB, and for real-world experiments, we utilized the Robot Operating System (ROS)[22] to connect to robots and sensors in our lab.

5.2 Matlab Simulations

This series of simulated experiments was conducted in MATLAB R2022B [15], on an Ubuntu 20.04 machine equipped with an AMD Ryzen 3800X processor, an NVIDIA 2080 Super GPU, and 32GB of RAM.

5.2.1 Simulated Sensing and Dynamics

The primary sensor used in our experiments is a simulated LiDAR built in MATLAB. This sensor is assumed to have perfect information within its field of view and is able to precisely measure the distance from each virtual laser to the surface it intersects with. The LiDAR is modeled with a 360° field of view and an 8m range. The robot's motion is modeled using non-holonomic unicycle dynamics in accordance with how we've chosen to design our MPC 4.5, and the dynamics of the obstacles in the environment are modeled as simplistic point model 4.6. In short, the dynamics of the entities in the simulation match the dynamics that the MPC was designed to model.

5.2.2 Visualization Overview

This section serves to explain the graphical representations and significance of each entity in our MATLAB simulations, which remain consistent throughout our simulated and real experiments. Figure 5.2 below shows a top-down view of one of our MATLAB simulations in operation. In the center of the scene, a robot being controlled by our system is represented by a blue triangle. The trajectory that the robot has followed so far is shown in blue, and the planned path resulting from the series of optimal controls produced by the MPC is represented by the dashed green line. The goal pose that the robot is trying to reach is represented by the orange dot. The robot is surrounded by a red circle, which represents a safety radius around the robot. Similarly, the bold, black circles in the robot's surroundings represent safety radii around obstacles in the environment. These radii are chosen arbitrarily in our simulated experiments, but during real-world testing these are derived from coarse measurements of the actual obstacles they represent. Objects that are too large or eccentric to be approximated by a single circle are instead approximated by a series of adjacent or overlapping circles.

Our MPC's obstacle avoidance constraints ensure that we will always maintain space no less than the sum of these two radii between our path and each obstacle, which is the foundation of how we are able to avoid them. In the discussion of our approach and experiments, we refer to this first black circle as an obstacle's initial position because it represents the initial position of the obstacle at the beginning of the MPC's current control horizon.

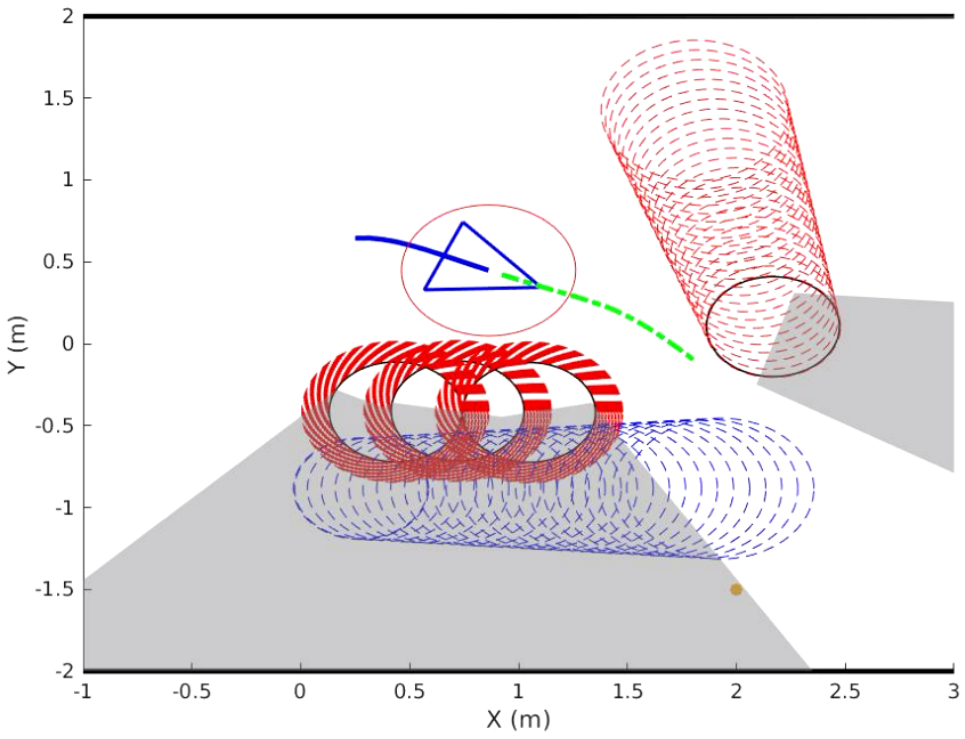


Figure 5.2: Overview of our MATLAB simulation environment

The dashed and colored circles emanating from each obstacle's initial position represent a prediction based upon that obstacle's last observed position and velocity, affected by gradual inflation to account for uncertainty present in both the movement of the robot and our ability to measure it. Red dashed circles represent predictions given the true, observed dynamics of a visible obstacle, and blue dashed circles represent predicted dynamics of an occluded obstacle. The trajectory rendered in blue represents an approximation of the 'worst-case path' that the occluded obstacle could take, which is the path that minimizes the amount of time the robot has to react to the occluded actor reappearing in the robot's field of view. Each successive circle represents the next position in a sequence of predicted positions that corresponds with each step of the MPC's horizon. It's also worth noting that these predictions may visibly overlap during these experiments, as the other entities in the environment are not constrained to respect the safety radii or each other.

Visibility, and the lack thereof, is represented by the dark grey and white area of the environment. The area shown in white is the region visible to the robot, and the dark grey regions represent the area outside of the robot's Field of View (FOV) or occlusions caused by obstacles blocking the robot's line of sight.

5.2.3 Simulation 1: Anticipating the Worst-Case Scenario

The first simulated case we explored required the robot to navigate to a goal while avoiding two dynamic actors. One of these actors was briefly visible to the robot but quickly moved out of view and remained occluded for the majority of the robot's mission. Of the scenarios tested during the development of our approach, we chose to include this one because of how clearly it illustrates the benefit of anticipating the worst-case scenario. We first tested a 'Naive' approach– the same baseline MPC that we designed for our approach, but without any of the attentive or proactive prediction capabilities. Figure 5.3 demonstrates how an occluded actor suddenly appearing in the robot's field of view can easily induce an erratic and potentially unsafe trajectory.

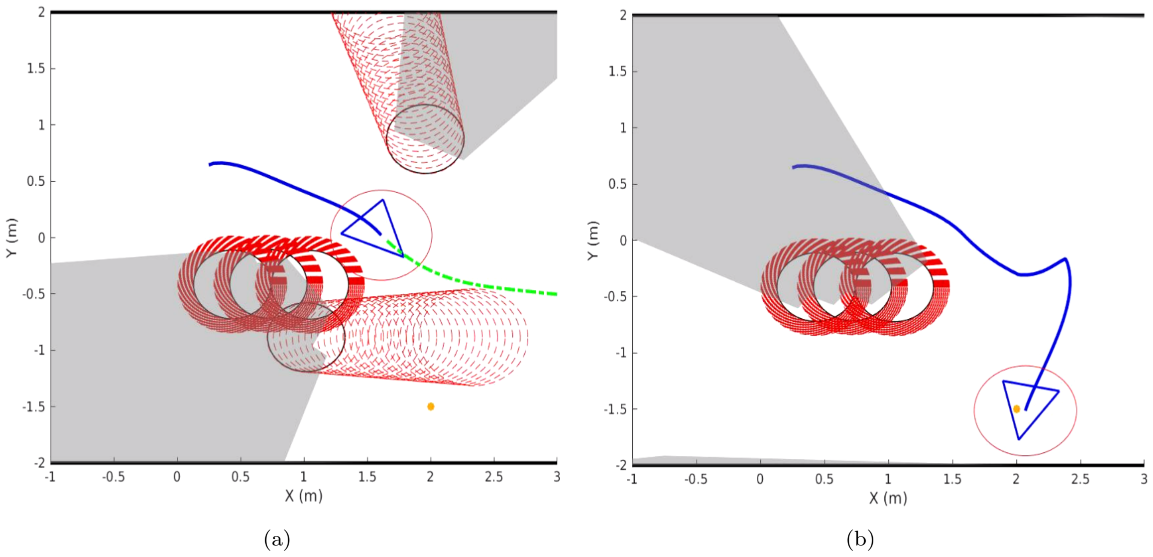
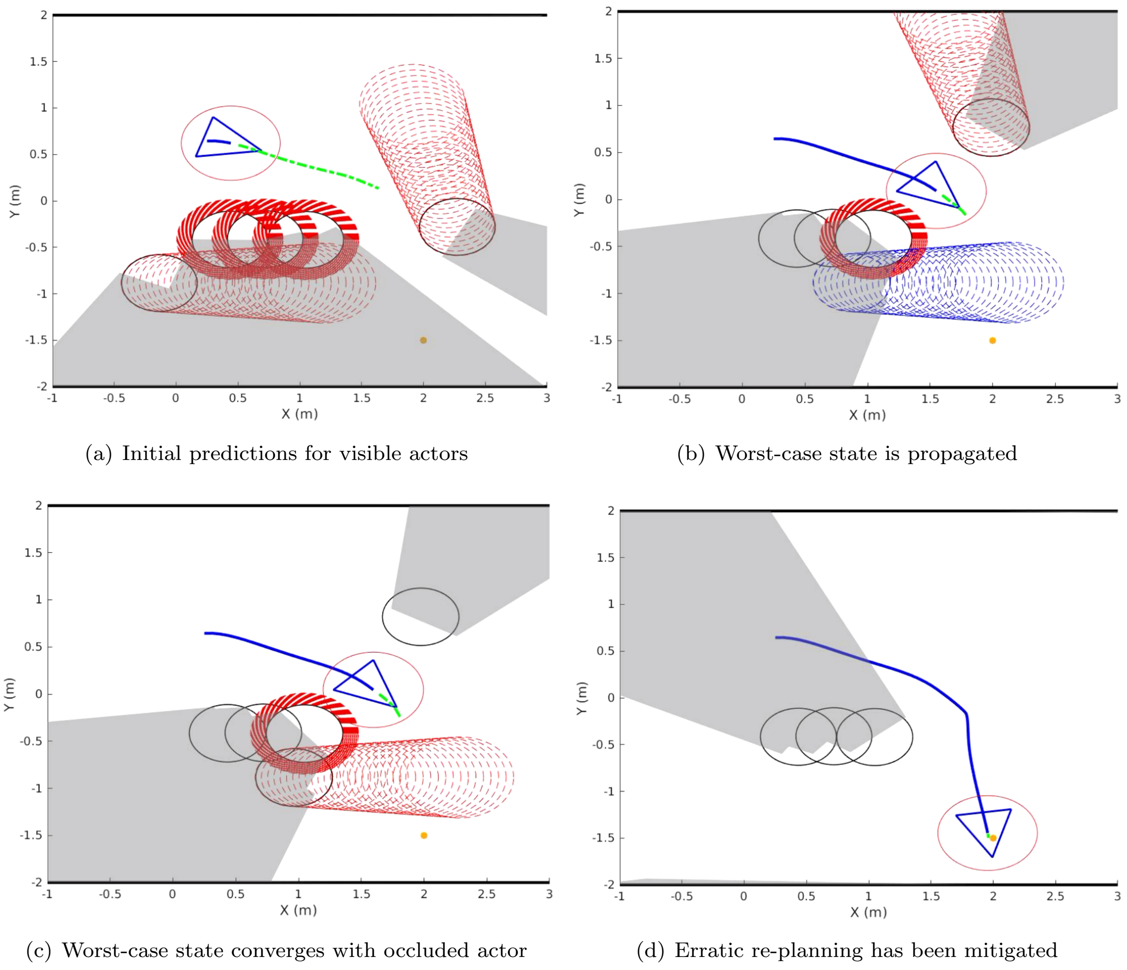


Figure 5.3: The Naive agent is caught off guard (a) and plans a harsh detour en route to the goal (b).

Next, we ran the same simulation with our full approach active. Figure 5.4(a) shows the progression of our approach's initial prediction converging to the worst-case scenario on the boundary of the occluded region, and Figure 5.4(d) demonstrates the benefit of that prediction. Because our approach considered the worst-case scenario in its motion planning optimization, our robot took a more conservative trajectory and avoided a drastic evasive maneuver.

We ran two additional permutations of the experiment with two more configurations of our approach: one only using the attentive selection component of our approach and another only using its' proactive prediction



(a) Initial predictions for visible actors

(b) Worst-case state is propagated

(c) Worst-case state converges with occluded actor

(d) Erratic re-planning has been mitigated

Figure 5.4: The robot correctly predicts the trajectory of an occluded actor and is able to anticipate its sudden re-appearance

component. This allowed us to get some additional perspective on how our approach was improving the robot's performance. The following series of figures compare the Naive, Attentive, Proactive, and Attentive-Proactive approaches in detail.

The benefits of the Proactive component of our approach are clear when we examine the robot's dynamics (Figure 5.5) during the course of the experiment. The Proactive and Proactive-Attentive approaches maintain a much more consistent and smooth velocity profile throughout the mission and achieve higher mean velocities as a result of that. The Proactive approaches also arrive at their goal pose over 20% faster than the Naive approach, which is a substantial increase in agility. The pure Attentive component of our approach did not have a noticeable effect on performance during this mission, but this aligns with our intuition. Since the actor that became occluded during the scenario was tracked by the Attention module, nothing was excluded from the set of dynamic predictions and thus no performance gain was realized.

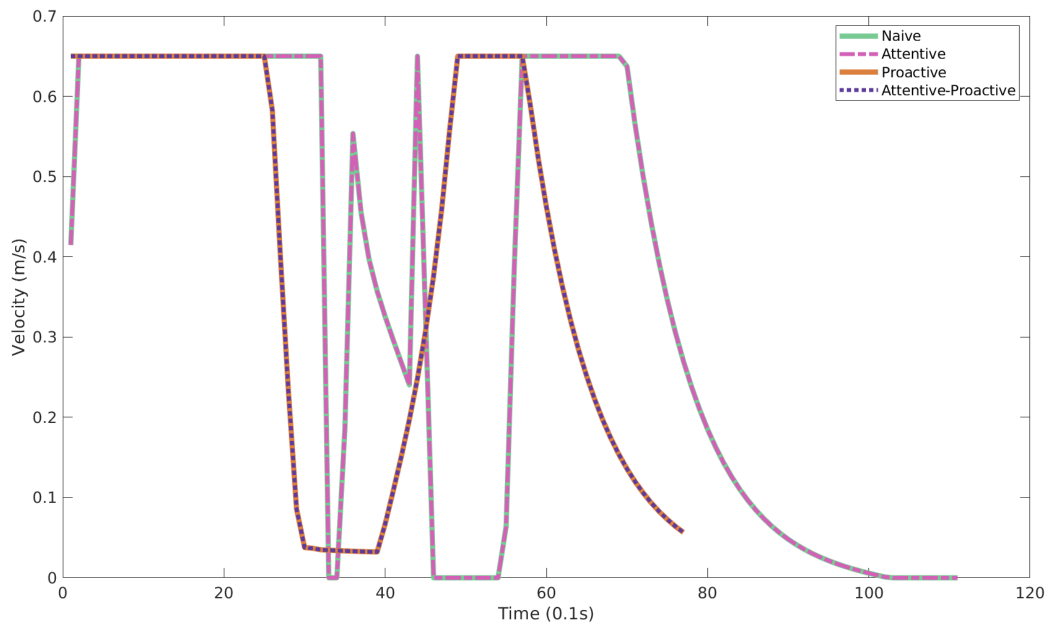


Figure 5.5: Comparison of the velocity profiles of different permutations of our approach

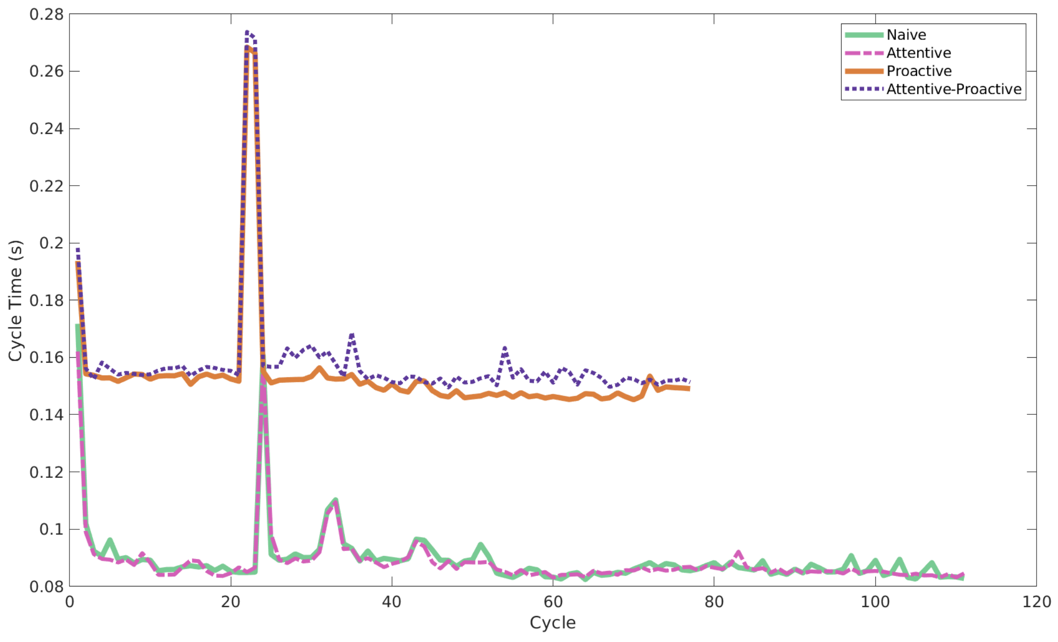


Figure 5.6: Performance comparison between different permutations of our approach

When we examine cycle times across the four approach permutations, as shown in Figure 5.6, we see that this enhanced agility comes at an increase in computational cost. The approaches utilizing the Proactive approach component have an average cycle time of 0.1588ms, which is a 71% increase in computational cost.

However, if the Attention component had been more active in this scenario, we would have expected it to help offset that cost increase.

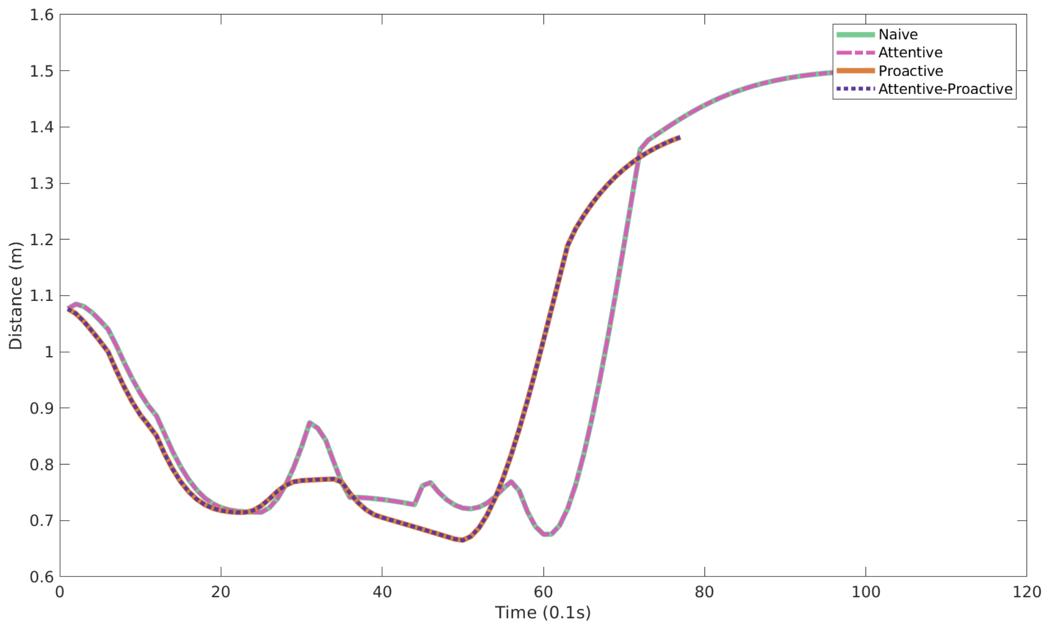


Figure 5.7: Distance to nearest actor comparison between different permutations of our approach

Next, we compare the Euclidean distance between the robot and the nearest obstacle over the course of the simulation (5.7). When running the simulation using the Naive approach, the mean Euclidean distance to the nearest obstacle $\overline{d}_e$ =1.0549m. However, when we run the simulation using the Attentive-Proactive approach, we measure $\overline{d}_e$ = 0.9017m. Figure 5.8 demonstrates why we believe this occurs. Since the robot has taken a conservative path, it begins the obstacle avoidance maneuver closer to the obstacle rather than becoming deflected by it. As soon as the robot perceives enough space between other obstacles to navigate through, it will attempt to do so with the tightest possible clearance that it can achieve. In this manner, a lower $\overline{d}_e$ indicates that the Attentive-Proactive configuration of the approach is superior in navigating precisely through crowded environments.

Finally, we examine the error between the robot's predicted and actual trajectories at each cycle. This is accomplished by comparing the predicted trajectory produced by the MPC-issued optimal control sequence at time t to the predicted trajectory at time $t-1$ and computing the Euclidean distance between each point in both trajectories. This produces a measure of how conservative the robot's predicted trajectories are over the course of a mission. A high trajectory prediction error suggests that the robot has had to compute a significantly different trajectory in response to sudden stimuli, and a low trajectory prediction error suggests

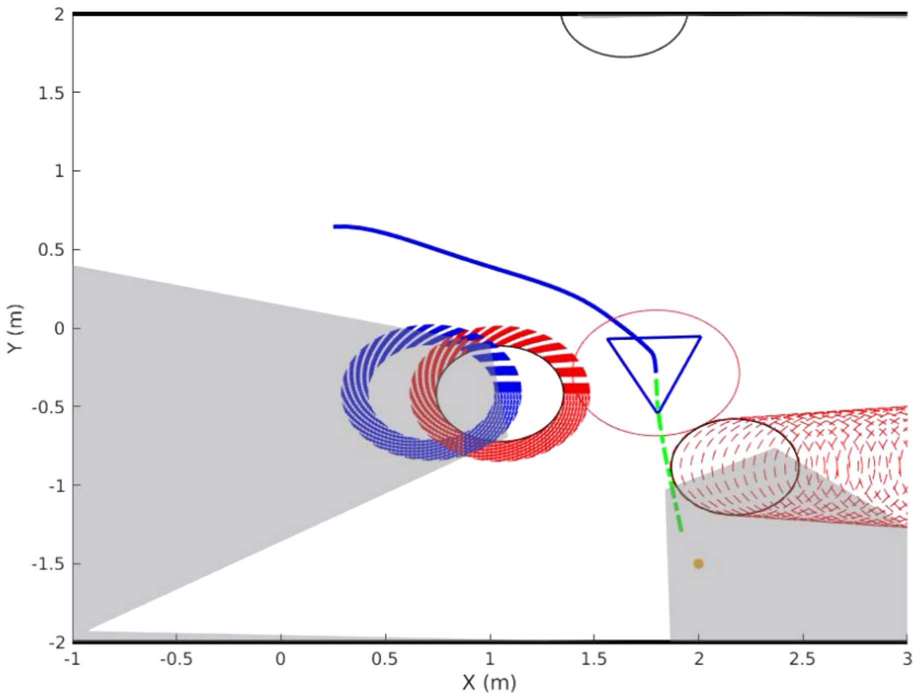


Figure 5.8: Rationale for how the Attentive-Proactive approach achieves a lower mean distance to the closest actor during the simulation

that the robot correctly anticipated the stimuli encountered over the course of its mission.

Figure 5.9 compares the trajectory prediction errors in each approach formulation over the course of Simulation 1. The Proactive and Attentive-Proactive approaches substantially outperform the Naive and Attentive approaches, maintaining a trajectory prediction error of less than 0.1m for the entirety of the mission. Of note, the configurations that leverage the Proactive approach avoided four major prediction deviations during this simulation, three of which exceeded a 0.5m deviation from the previous predicted trajectory.

To summarize our findings of the first simulation, we observed that our full approach was able to significantly improve overall agility and resilience to disturbance from the optimal path in exchange for an increase in computational expense. In subsequent experiments, we hope to explore how engaging the attention component of our approach can help discount the expense of this benefit.

5.2.4 Simulation 2: Occlusions in a Crowd

Our second simulated case required the robot to navigate through a crowd of dynamic actors on its way to a goal. While the environment appeared open enough to navigate through at the beginning of the simulation, the actors quickly converged into a chaotic tangle directly in front of the robot's goal, Figure 5.10. We chose to include this scenario to illustrate the benefit of predicting occluded behavior in dense crowds.

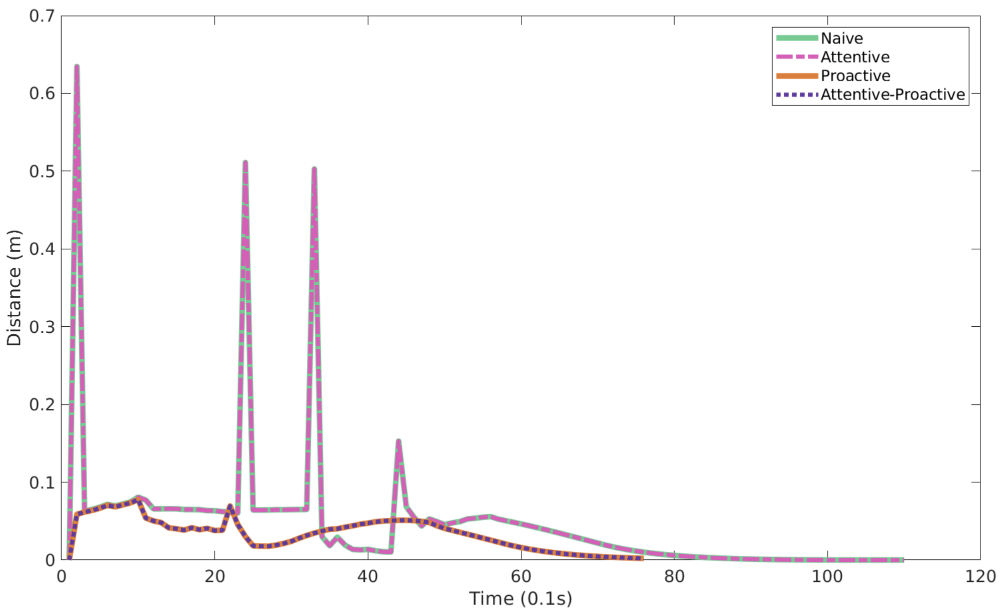


Figure 5.9: Error between the robot's predicted and actual trajectory during Simulation 1

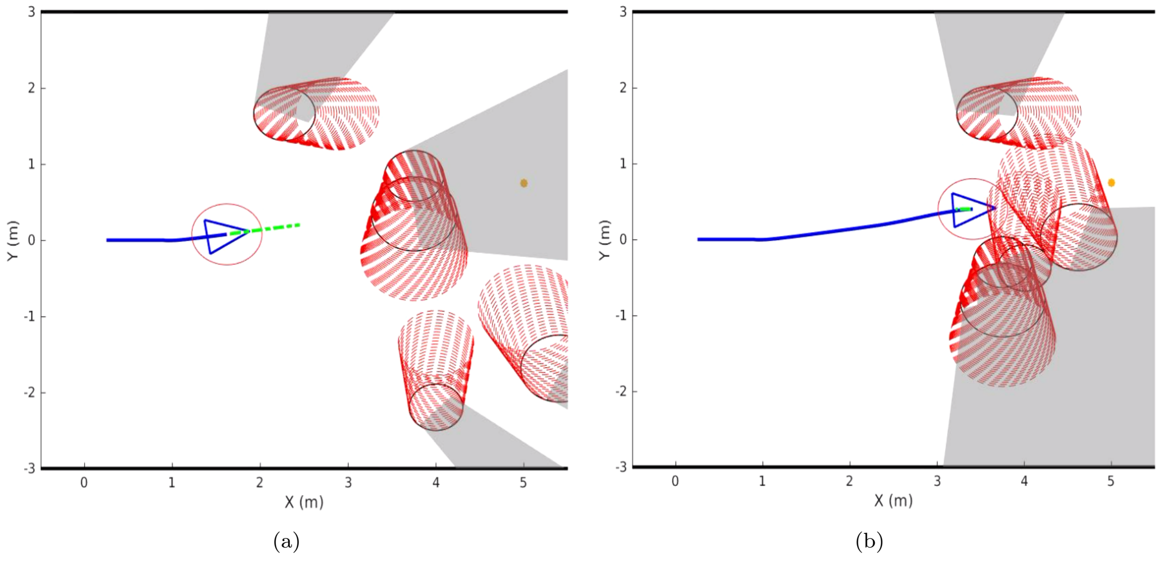


Figure 5.10: Initial state of the crowd (a) and crowd converging, with actors beginning to occlude each other (b).

Figure 5.10(b) shows results similar to the first experiment, where the two previously occluded actors suddenly reappeared in close proximity to the robot. This resulted in the Naive approach producing another erratic maneuver. However, activating the Attentive-Proactive approach during the next attempt produced a smoother and more stable path to the robot's goal. Because our approach was able to anticipate the potential worst-case scenarios for those occluded actors, by the time they reappeared the robot was able to smoothly

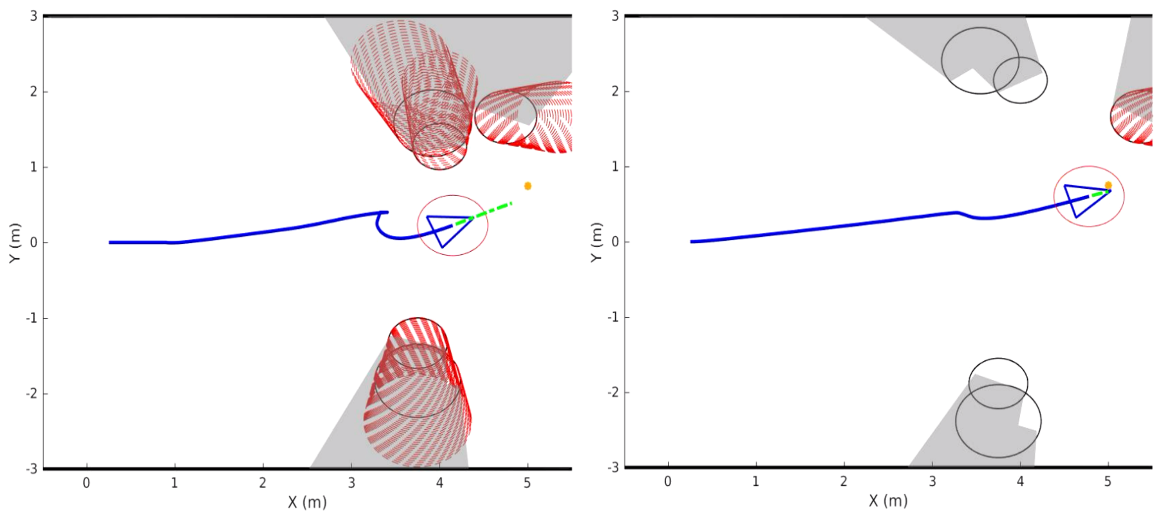


Figure 5.11: The Naive approach is unable to smoothly navigate the crowd and produces an erratic maneuver (a) but the Attentive-Proactive plans a smooth, safe path (b).

decelerate and steer around them. 5.11(b)

Next, we re-ran this test with the different permutations of our approach to get a closer look at its performance.

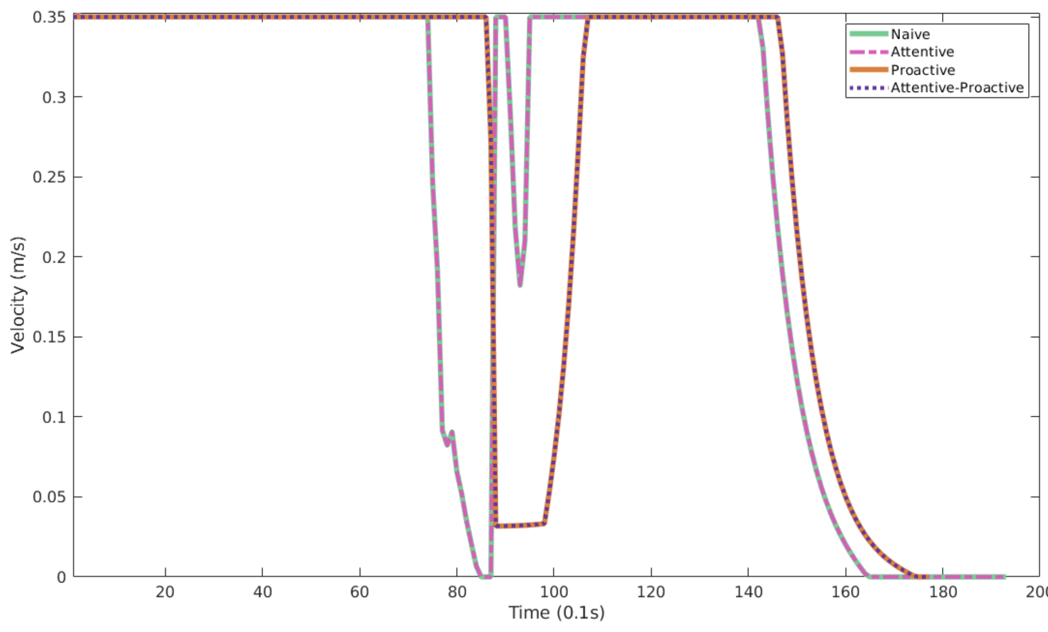


Figure 5.12: Comparison of the velocity profiles of different permutations of our approach

As shown in Figure 5.12, the Attentive-Proactive planner was able to outmaneuver the Naive and pure

Attentive planners and arrive at the goal almost two seconds earlier. Observing the Naive approach's behavior shows that it takes approximately two seconds to return itself to the optimal trajectory after being perturbed by an unknown occluded actor. The time cost of this sort of detour is significant, especially in environments where this sort of occurrence can be encountered multiple times in a single mission.

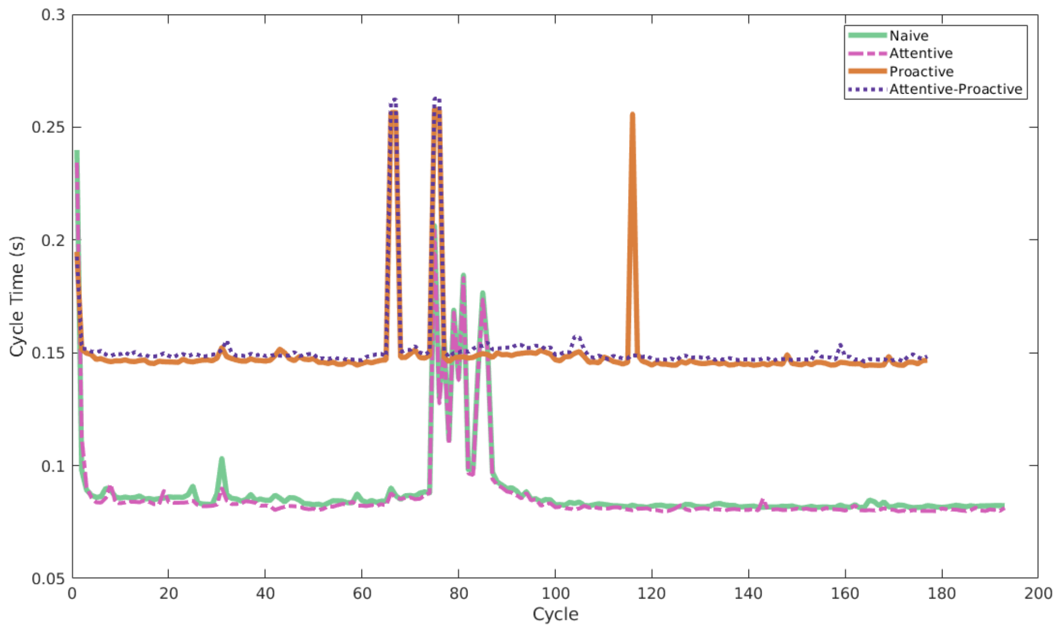


Figure 5.13: Performance comparison between different permutations of our approach

The performance metrics for this simulation closely matched the behavior observed in the previous simulation; the Attentive-Proactive approach is roughly 70% more computationally expensive than the Naive approach. However, in this scenario we also observed an example of the Attentive Actor Tracking module discarding a prediction, which avoided an approximately 0.1s cycle time spike. A plot of these metrics is shown in Figure 5.13.

Comparing the Euclidian distance between the robot and the nearest obstacle over the course of the mission (Figure 5.14) shows that the Attentive-Proactive approach outperforms the Naive approach by maintaining a more conservative margin around obstacles.

Finally, we examine the predicted trajectory error over the course of Simulation 2. Similar to what was observed in Simulation 1, we found that the configurations leveraging the Predictive approach maintained a more consistent and lower predicted trajectory error. Figure 5.15 shows the error between the robot's predicted and actual trajectories at each cycle in Simulation 2.

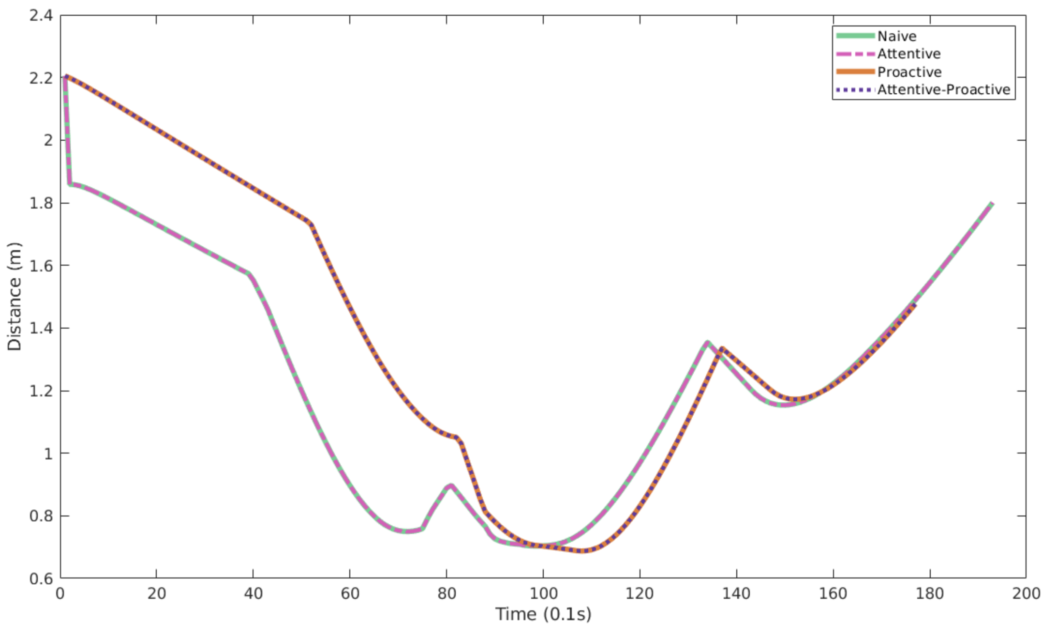


Figure 5.14: Distance to nearest actor comparison between different permutations of our approach

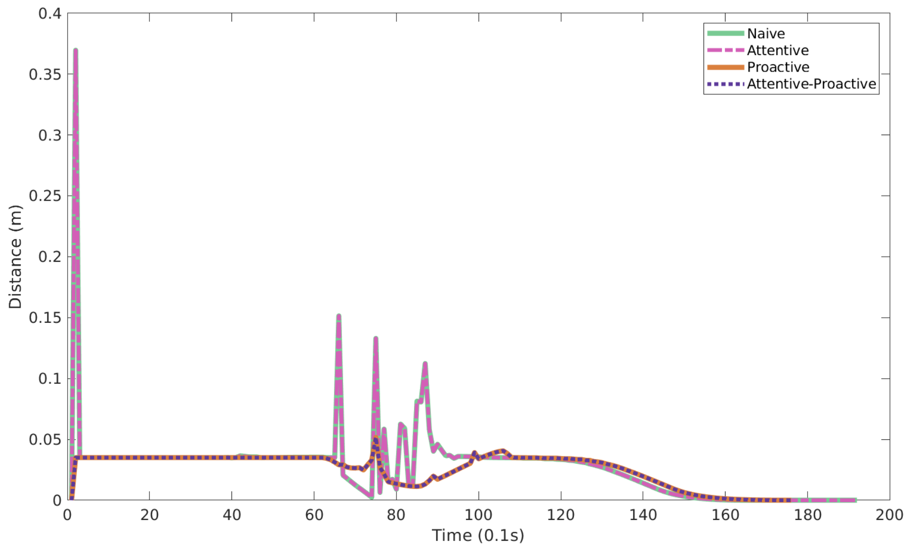


Figure 5.15: Error between the robot's predicted and actual trajectory during Simulation 2

5.3 Experiments

In addition to evaluating our approach in a simulated environment, we also performed a real-world experiment in the Autonomous Mobile Robotics Lab[3] at the University of Virginia. The experiment was designed to be analogous to our first simulation, so that we could assess the capabilities of our approach on a real hardware platform.

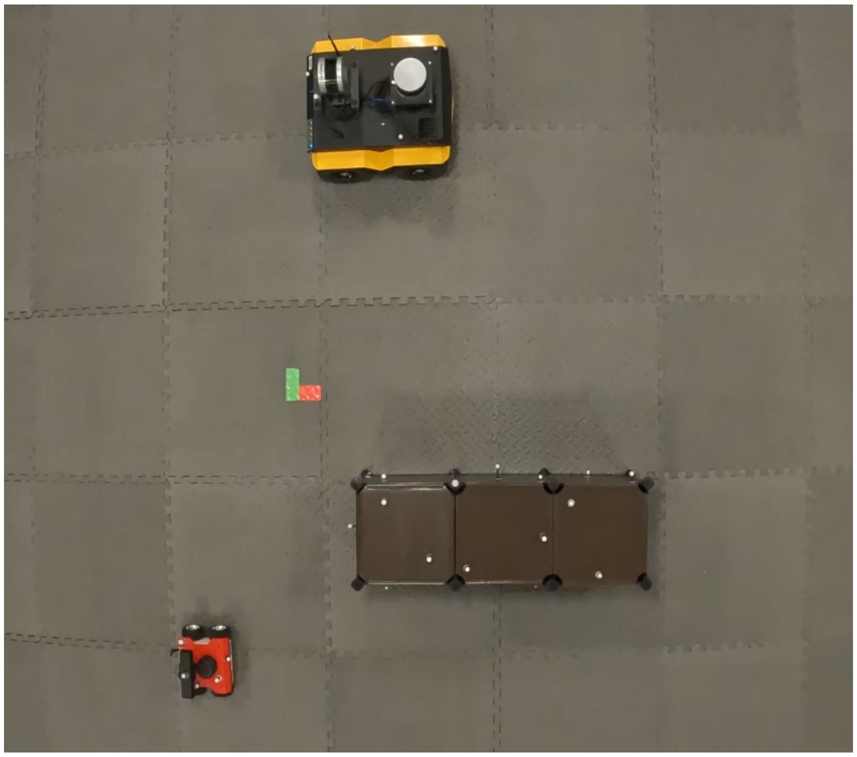


Figure 5.16: Overhead view of our experiment setup

We constructed a scenario for the experiment consisting of two small Unmanned Ground Vehicles (UGVs) and three small barrier objects to provide occluding features in the environment (Figure 5.16). Our primary test platform was a Clearpath Jackal [21], a small, non-holonomic UGV purpose-built for robotics research applications. In addition, we also utilized a ROSBot 2[14], another small robotics platform for learning and development, for use as an additional autonomous actor for our tests. With this experiment, we sought to replicate the results of our MATLAB simulations in the physical world. Figure 5.17 shows a plot of the robot and its environment during the experiment,

After testing both the Naive and Attentive-Proactive approaches on the Jackal, we found that both performed similarly to how they performed in simulations. When the occluded actor reappeared into view of the robot controlled by the Naive approach, it planned a reactive detour away from the goal that almost took the robot outside the boundaries of the test area. However, just like in the simulation, the Attentive-Proactive approach was able to anticipate the worst-case trajectory of the occluded actor. Figure 5.18 compares the planned trajectory of the Naive approach with the Attentive-Proactive approach during the experiment, illustrating how the Naive approach planned a detour around the actor and the Attentive-Proactive approach slowed down to let it pass by. Photographs of the experiment at this moment are shown in Figure 5.19 showing the results of these differently planned trajectories in the physical world. These images have been annotated to provide additional context to the scene. The robot's planned trajectory is annotated in yellow, and actor obstacle avoidance radii are annotated in red. Since the robot had already been planning around the occluded actor's worst-case trajectory prior to its reappearance, it was able to maintain its smooth trajectory toward the goal.

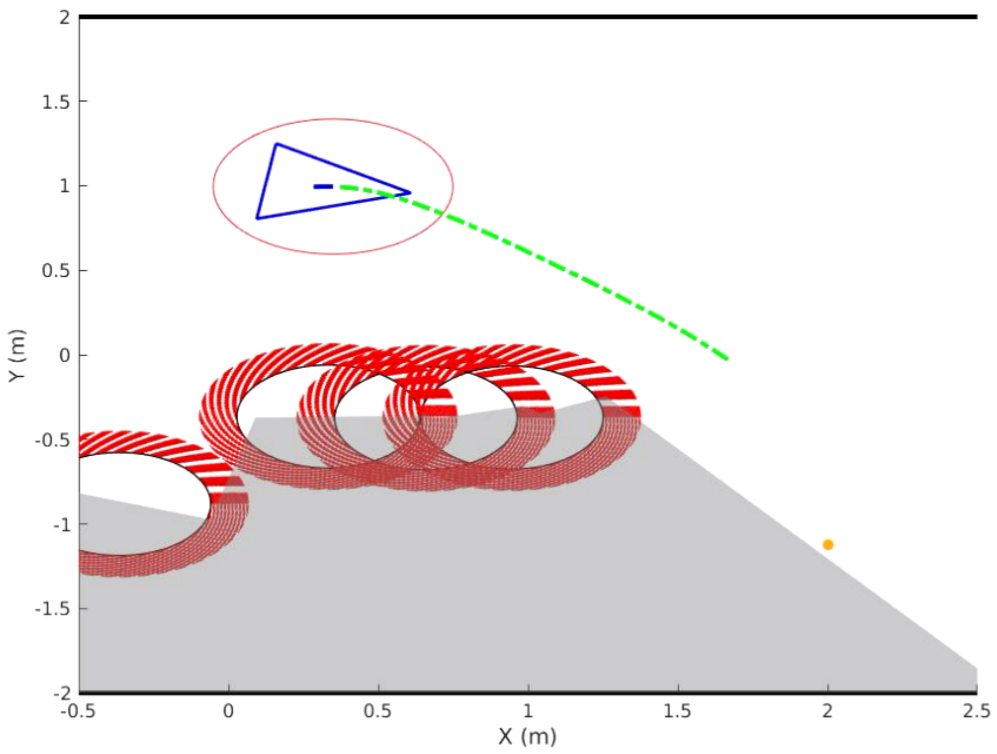
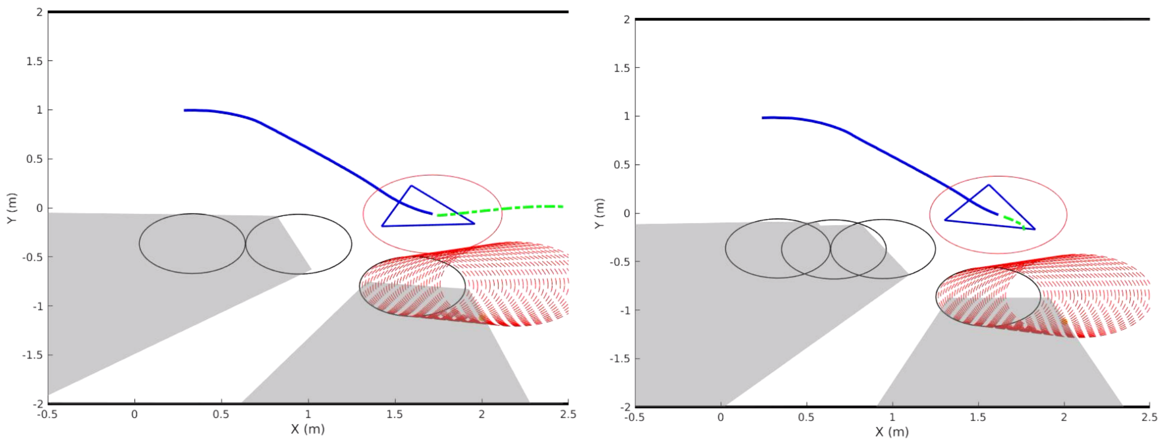
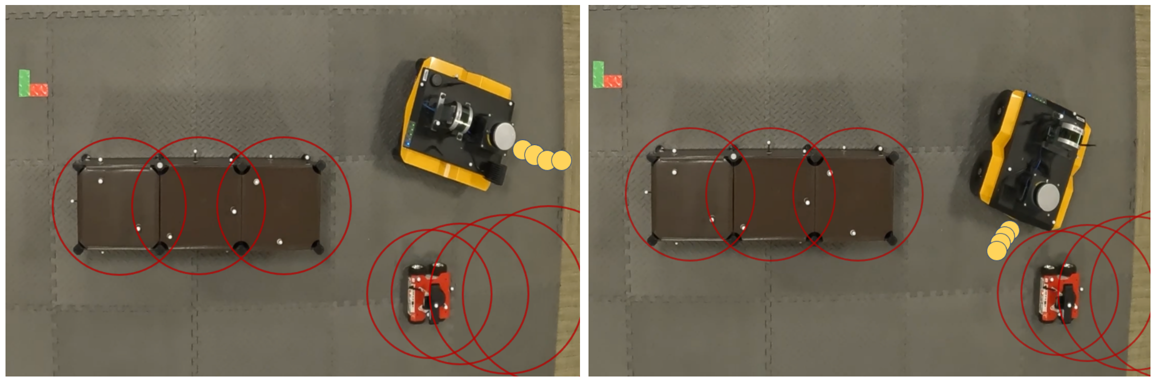


Figure 5.17: Initial state of the experiment, rendered with obstacle radii



(a) (b)

Figure 5.18: Comparing paths generated by the Naive (a) and Attentive-Proactive (b) in the experiment



(a) (b)

Figure 5.19: Obstacle avoidance comparison between the Naive (a) and Attentive-Proactive (b) approaches during a real-world experiment

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this thesis, we have presented a novel synthesis of motion planning methods that allows us to proactively avoid dynamic obstacles with intermittent visibility in crowded environments. To achieve this, we first introduced a method to monitor obstacle state evolution over time and to make inferences about their intended trajectories from those observations. These inferences allowed us to determine which obstacles were important to pay attention to, which in turn minimized the number of obstacles we needed to consider for subsequent motion planning optimizations. Next, we utilized our obstacle state observations to make predictions about the future trajectories of the significant obstacles. These trajectories allowed us to construct a sequence of future positions for these obstacles, which were used as constraints by an MPC to generate optimal collision avoidance behavior. We validated this approach through simulations and real-world experiments and found that it was able to reliably produce safe trajectories that were resilient to collision with intermittently occluded obstacles. These experiments demonstrated that our approach produced conservative predictions that remained consistent from cycle to cycle, with minimal deviation in response to sudden stimuli. Furthermore, we found that our approach outmaneuvered a traditional, naive MPC in crowded and occlusion-rich scenarios, increasing agility by as much as 20% under some conditions. However, one major limitation of this capability is that it comes at the expense of increased computation time. This is acceptable for a prototype, but in order to deploy this in a real-world capacity we will need to find methods of further reducing that computational expense. One immediate way we could address this would be to explore porting our software to a better-optimized programming language. The more we're able to optimize our software stack, the better we'll be able to tune our performance for real-world applications. This limitation and our plans to mitigate it will be discussed further in the 'Future Work' section of this report. We consider these to be very promising initial results and are eager to continue exploring this problem space in future work.

6.2 Future Work

During the development of our approach, we identified several areas of interest that we would like to explore further in our future work.

6.2.1 Software Package Improvements

As one of our short-term goals for this work, we plan to port our code to Python and open-source it to the broader academic community. While MATLAB provided a straightforward and troubleshooting-friendly development environment during the bulk of our work, it lacks the accessibility and ubiquity necessary for our work to be easily shared with others. Fortunately, both ROS and Casadi have native support for Python, which will eliminate a lot of potential friction when trying to port our code to a new language.

We also plan to explore a deeper optimization of our software architecture. Spending some time analyzing the code with a performance profiler will allow us to identify key areas of the code where we can improve our performance, and improving those areas will further improve the quality of our results. Notably, since the performance of certain functions can vary from language to language, we plan to explore this after porting the code to Python.

6.2.2 Enhanced Actor Trajectory Sampling

One of our approach's core assumptions was that we were able to compute a 'worst-case scenario' trajectory for tracked and occluded actors given the geometry of the environment and the obstacles within it. While this was black-boxed to limit the scope of this research, we acknowledge that it could be a fairly deep area of study in its own right. We plan to explore this topic in greater depth in the future, and try to develop an enhanced actor trajectory sampling module to improve the capabilities of our system further. One potential approach to this that we're beginning to explore is using an adversarial path planner to determine the most aggressive 'intercept course' an occluded actor might take to disrupt our planned path. The occlusion boundary convergence behavior will likely continue to function in the same way, but with the added advantage of the trajectories not needing to be defined a priori.

Another approach we would like to explore integration with is probabilistic trajectory prediction. Probabilistic approaches generate a large number of potential trajectories for sensed actors and assign a probability to describe the likelihood that the actor follows each possible trajectory. This probability score can serve as an additional data point for determining the salience of actors in the robot's vicinity. In addition to enhancing our approach's ability to determine actor salience, this style of trajectory prediction could be very interesting to apply to salient occluded actors in order to predict more complex worst-case scenario behaviors. Furthermore, leveraging probabilistic trajectory prediction could augment the trajectories we predict for salient occluded actors beyond just the worst-case scenario. As an addition or replacement to predicting the worst-case scenario trajectory, we could also explore predicting the most probable trajectory that the occluded actor could follow.

6.2.3 Further Testing and Design Iteration

We plan to test our approach against comparable and competitive motion planning approaches in order to better assess our performance against the state of the art. In addition to understanding how our solution performs relative to other common solutions, this may also allow us to draw insight from other approaches and find new ways to improve our system further. We also plan to test our approach on a broader variety of robotic platforms. The systems we initially integrated with (Clearpath Jackal, ROSbots) were compatible with the non-holonomic unicycle model we initially developed with, but we are very interested in testing our approach out on more complex dynamical systems like quadrupedal or bipedal robots.

Bibliography

[1] R. Alami, T. Simeon, and K. Madhava Krishna. "On the influence of sensor capacities and environment dynamics onto collision-free motion plans". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. 2002, 2395–2400 vol.3. DOI: 10.1109/IRDS.2002.1041626.

[2] Joel A E Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* (2018).

[3] Nicola Bezzo. *AMR Lab Website*. URL: https://www.bezzorobotics.com/r.

[4] Maxime Bouton et al. "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.

[5] Engin Bozkurt. *Lidar Obstacle Detection*. URL: https://github.com/enginBozkurt/LidarObstacleDetection/tree/master.

[6] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 392–399. DOI: 10.1109/ITSC.2014.6957722.

[7] Shuo Cheng et al. "Longitudinal Collision Avoidance and Lateral Stability Adaptive Control System Based on MPC of Autonomous Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 21.6 (2020), pp. 2376–2385. DOI: 10.1109/TITS.2019.2918176.

[8] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2010. ISBN: 9781139855594. URL: https://books.google.com/books?id=0cggAwAAQBAJ.

[9] Mohamed Elbanhawi and Milan Simic. "Sampling-Based Robot Motion Planning: A Review". In: *IEEE Access* 2 (2014), pp. 56–77. DOI: 10.1109/ACCESS.2014.2302442.

[10] Dashan Gao and Nuno Vasconcelos. "Discriminant Saliency for Visual Recognition from Cluttered Scenes". In: *Advances in Neural Information Processing Systems*. Ed. by L. Saul, Y. Weiss, and L. Bottou. Vol. 17. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper_files/paper/2004/file/dda04f9d634145a9c68d5dfe53b21272-Paper.pdf.

[11] Jacob Higgins and Nicola Bezzo. "A Model Predictive-based Motion Planning Method for Safe and Agile Traversal of Unknown and Occluding Environments". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 9092–9098. DOI: 10.1109/ICRA46639.2022.9811717.

[12] Jacob Higgins and Nicola Bezzo. "Negotiating Visibility for Safe Autonomous Navigation in Occluding and Uncertain Environments". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4409–4416. DOI: 10.1109/LRA.2021.3068701.

[13] Pei-En Hsu et al. "Collision Avoidance for AGV Navigation Using the Time Elastic Band Algorithm". In: *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*. 2023, pp. 855–856. DOI: 10.1109/ICCE-Taiwan58799.2023.10227064.

[14] Hussarion.com. *Rosbot landing page*. URL: https://store.husarion.com/collections/robots/products/rosbot.

[15] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: https://www.mathworks.com.

[16] Jie Ji et al. "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints". In: *IEEE Transactions on Vehicular Technology* 66.2 (2017), pp. 952–964. DOI: 10.1109/TVT.2016.2555853.

[17] Steven M. LaValle. "Motion Planning". In: *IEEE Robotics & Automation Magazine* 18.1 (2011), pp. 79–89. DOI: 10.1109/MRA.2011.940276.

[18] Keyu Li et al. "SARL: Deep Reinforcement Learning based Human-Aware Navigation for Mobile Robot in Indoor Environments". In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 688–694. DOI: 10.1109/ROBIO49542.2019.8961764.

[19] Pinxin Long et al. "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6252–6259. DOI: 10.1109/ICRA.2018.8461113.

[20] R. Peddi. "Interpretable Monitoring for Self and Socially Aware Mobile Robot Planning". PhD thesis. University of Virginia, 2022.

[21] Clearpath Robotics. *Clearpath Robotics Jackal Page*. URL: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/.

[22] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: https://www.ros.org.

[23] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. "Safety-Critical Control and Planning for Obstacle Avoidance between Polytopes with Control Barrier Functions". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 286–292. DOI: 10.1109/ICRA46639.2022.9812334.

[24] Maria Tsiourva and Christos Papachristos. "LiDAR Imaging-based Attentive Perception". In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2020, pp. 622–626. DOI: 10.1109/ICUAS48674.2020.9213910.

[25] Allan Wang, Christoforos Mavrogiannis, and Aaron Steinfeld. "Group-based Motion Prediction for Navigation in Crowded Environments". In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, Aug. 2022, pp. 871–882. URL: https://proceedings.mlr.press/v164/wang22e.html.

[26] L.R. Williams and D.W. Jacobs. "Stochastic completion fields: a neural model of illusory contour shape and salience". In: *Proceedings of IEEE International Conference on Computer Vision*. 1995, pp. 408–415. DOI: 10.1109/ICCV.1995.466910.

[27] Michael T. Wolf and Joel W. Burdick. "Artificial potential functions for highway driving with collision avoidance". In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 3731–3736. DOI: 10.1109/ROBOT.2008.4543783.