Autonomous Navigation Among Dynamic Obstacles

by

Himanshu Gupta

B.Tech., Indian Institute of Technology Ropar, 2017

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2022

Committee Members:

Dr. Bradley Hayes, Chair

Dr. Zachary Sunberg

Dr. Lijun Chen

Gupta, Himanshu (M.S., Computer Science)

Autonomous Navigation Among Dynamic Obstacles

Thesis directed by Assistant Professor Dr. Bradley Hayes

Existing approaches to autonomy can handle highly structured environments such as confined spaces within factories or lanes on freeways. However, they are still uncommon in unstructured settings with uncertain dynamic agents in the environment. In this work, we present a hybrid online Partially Observable Markov Decision Process (POMDP) planning system that addresses the problem of autonomous navigation in the presence of multi-modal uncertainty introduced by other agents in the environment. As a particular example, we consider the problem of autonomous navigation in dense crowds of pedestrians and among obstacles. One previous approach to this problem first generates a path using a complete planner (e.g., Hybrid A*) with ad-hoc assumptions about uncertainty, then use online tree-based POMDP solvers to reason about uncertainty with control over a limited aspect of the problem (i.e. speed along the path). We present a more capable and responsive real-time approach enabling the POMDP planner to control more degrees of freedom (e.g., both speed AND heading) to achieve more flexible and efficient solutions. This modification greatly extends the region of the state space that the POMDP planner must reason over, significantly increasing the importance of finding effective roll-out policies within the limited computational budget that real-time control affords. Our key insight is to use multi-query motion planning techniques (e.g., Probabilistic Roadmaps or Fast Marching Method) as priors for rapidly generating efficient roll-out policies for every state that the POMDP planning tree might reach during its limited horizon search. Our proposed approach generates trajectories that are safe and significantly more efficient than the previous approach, even in densely crowded dynamic environments with long planning horizons.

Acknowledgements

This work would not have been possible without the constant support and guidance from my advisors, Dr. Bradley Hayes and Dr. Zachary Sunberg. I want to thank them for giving me the freedom and time to pursue this work and for being patient with me. I also want to express my gratitude to my labmates, Carl Mueller and Shivendra Agrawal for meaningful discussions. I would also like to thank Andrew Mills for useful insights into path planning with Fast Marching Methods. Lastly, I want to thank my family and friends for their constant love and support.

Contents

Chapter

List of Tables

Table

List of Figures

Figure

Chapter 1

INTRODUCTION

It is increasingly common to find autonomous systems operating successfully in relatively predictable and structured scenarios. For instance, vehicles can drive autonomously to stay within their lane on freeways (Fig. 1.1). Robotic manipulators can also be seen operating autonomously in structured confined spaces inside factories (Fig. 1.2). However, it is still uncommon to see autonomous systems in unstructured environments with uncertain dynamic obstacles. Despite ample investments, more complex navigation tasks with less structure imposed on the dynamic elements remain open challenges [26, 34]. Interaction with other agents in the environment is a particularly prolific source of difficult problems. Navigating through a crowd of pedestrians is one important example of this. In science fiction movies like Star Wars, droids move deftly between the people walking around them, and intuitively pedestrians should not greatly impede a properly-controlled robot's motion. In order to choose a good trajectory, however, the robot must reason about the intentions of the humans around it, a task fraught with uncertainty.

The Partially Observable Markov Decision Process (POMDP) is a mathematical framework for optimal decision making in the presence of various types of uncertainty. Previous approaches to tasks like pedestrian navigation have used the POMDP framework for navigation-centric tasks (e.g. [7, 4, 10, 47, 27, 33]). However, in these approaches POMDP planning is often relegated to a very limited role (e.g. speed control), or to a very limited class of uncertainty (e.g. Gaussian or unimodal distributions [4, 10]). For instance, Bai et al. (2015) [7] use the hybrid A^* algorithm [17] to plan a drivable path from the vehicle's current position to its goal location, using a POMDP formulation

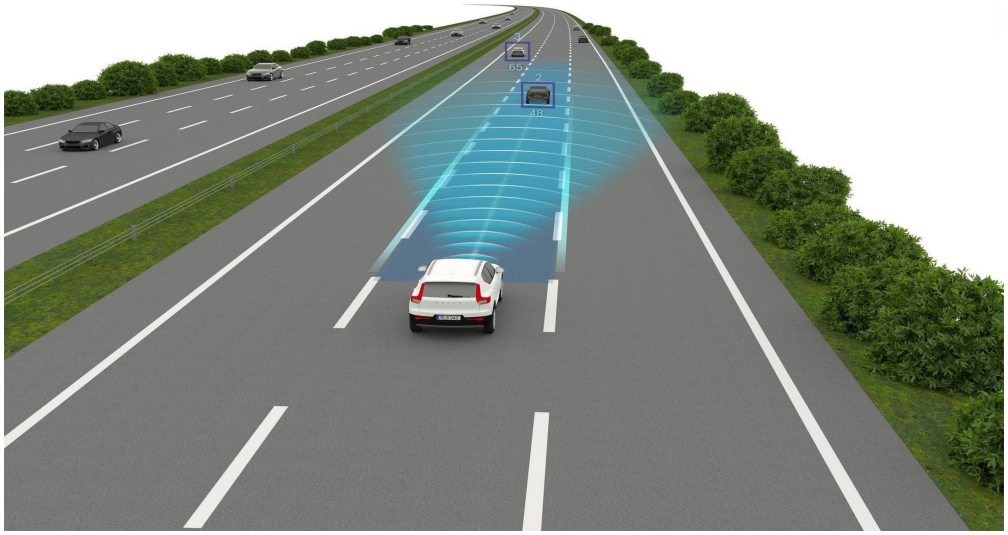

Figure 1.1: An autonomous vehicle using sensor observations to drive and stay in its lane. Image borrowed from [2]

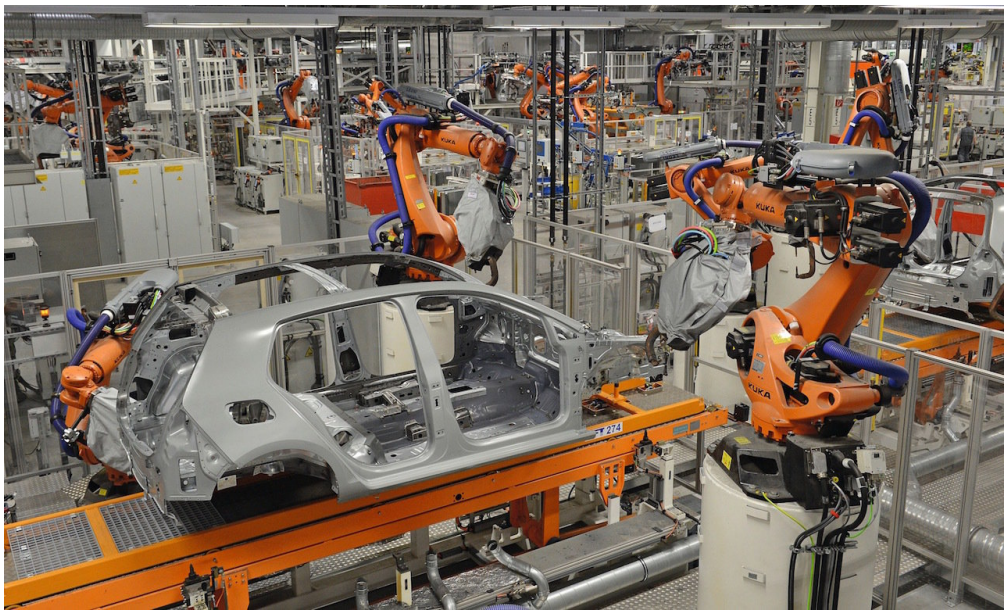

Figure 1.2: Robotic manipulators assembling a car in a structured factory setting. Image borrowed from [3].

only for speed control over that path. The observed reticence within the field to use POMDPs for large planning problems is understandable; in general obtaining exact solutions to POMDPs is an intractable problem [37]. However, sparse tree-based online planners are surprisingly insensitive to the size of the state and observation space [25, 43, 52, 46], suggesting a way forward for increasingly expressive POMDP formulations to improve the state of the art in critical control problems.

In contrast to the partitioned approach, we propose a more effective and general approach to planning for real-time tasks involving navigation with partial observability and arbitrary distributions. We assert that the POMDP planner should have control over all degrees of freedom that are relevant to the uncertainty it is facing to maximize its ability to generate satisfactory plans. For example, when navigating among pedestrians, the POMDP planner should have control over the speed **and heading**, rather than solely speed along a predetermined path [9, 22, 21, 7]. One may use online POMDP algorithms (e.g., DESPOT [52]) that perform a tree search guided by value estimates obtained by executing a roll-out policy.

Since an expansion of the action space can open up a much larger region of the state space to exploration, a critical challenge is determining a good roll-out policy for the vastly increased set of states reachable in the tree search. Since previous comparable approaches plan only along a 1-D path generated via A^*, roll-out policies are needed only for that single path, which are straightforward to specify by hand [7]. In the absence of an effective roll-out policy, a limited-horizon planner might never find the sparse positive terminal rewards that are typical in navigation tasks. The proposed method addresses this by incorporating multi-query motion planning techniques to produce a more informed roll-out policy, allowing for a corresponding increase in POMDP complexity and thus solution quality. To demonstrate the effectiveness of this approach we evaluate it with two motion planning methods, Probabilistic Roadmaps (PRM) and Fast Marching Methods (FMM), to generate effective roll-out policies.

Our evaluation shows that online navigation solutions to a POMDP with an extended action space and roll-out policies informed by multi-query planning methods are considerably more efficient in densely crowded environments than the two-step approach proposed in [7] without compromising

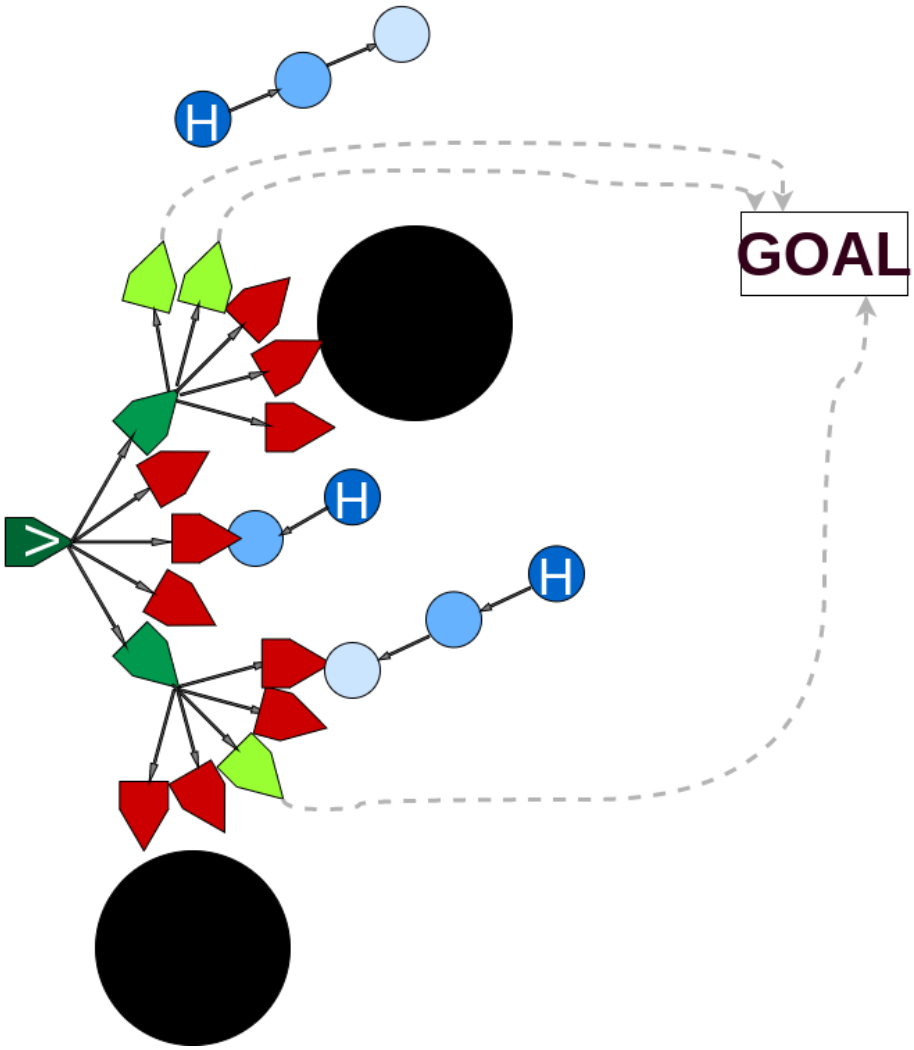

Figure 1.3: Two-dimensional POMDP motion planning with pedestrians. Green and red objects represent nodes in the planning tree, with green indicating high value. Blue circles denote the position of humans at different times. Black circles denote static obstacles. Dashed lines represent roll-out trajectories, a critical part of the proposed approach.

the safety of the pedestrians. The proposed approach explores multiple possible paths over a wider search space while reasoning over uncertainty in pedestrian intention as can be seen from Fig. 1.3, instead of handling uncertainty over just one path [7, 9, 22, 21]. We have also shown that the choice of multi-query planning technique does not affect the performance significantly, as long as it can generate an effective roll-out policy. Using pedestrian navigation as a motivating example for our proposed method throughout the remainder of this work, we refer to the popular unidimensional speed-based POMDP control formulation as Limited Space planner or LS planner, and our higher-dimensional (speed and heading) action space POMDP formulation as Extended Space planner or ES planner.

1.1 Thesis Outline

This chapter in intended to give a overview of the problem and the proposed approach to solve it. The thesis is structured in the following way, starting from the next chapter:

- Chapter 2: RELATED WORK - This chapter briefly explains the prior work on solving the problem of autonomous navigation among crowd.
- Chapter 3: TECHNICAL APPROACH - This chapter formalizes the problem of autonomous navigation among crowd as a POMDP and describes other essential technical components required for solving the problem.
- Chapter 4: EXPERIMENTS - This chapter describes in detail the simulation environment, different experimental scenarios and the different approaches that were implemented and compared.
- Chapter 5 : RESULTS and DISCUSSION - This chapter summarizes the results from our experiments and provides an explanation for the the obtained results.
- Chapter 6 : CONCLUSION and FUTURE WORK - This chapter concludes and mentions a few future directions of research.

Chapter 2

RELATED WORK

In recent years, a number of research efforts have focused on solving the problem of autonomous navigation in dynamic environments, especially among pedestrians.

2.1 Pedestrian Modeling

For safe and efficient navigation, a controller should incorporate pedestrian intentions and the corresponding behaviors into decision making. This raises the need for accurate models of pedestrian intention and behavior. A considerable amount of work has focused on using recorded trajectories to learn pedestrian dynamics [15, 5, 35]. However, these methods generally have large data requirements, and the learned model may not generalize well to new conditions. Since these learned patterns generally do not change after their generation, Vasquez et al. [51] presented an approach where motion patterns can be learned incrementally, and in parallel with prediction using Hidden Markov Models. Luo et al.[33] designed a pedestrian motion model that accounts for both intentions and interactions to capture pedestrian motions accurately.

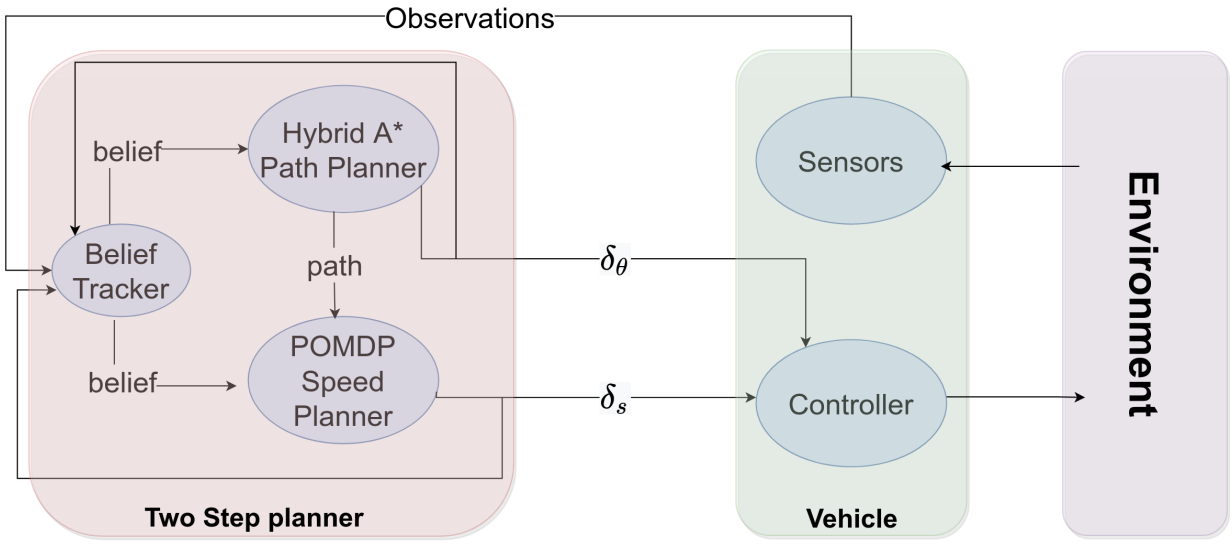

Figure 2.1: Two step POMDP-based planning for autonomous driving.

2.2 Vehicle Controller

Our work can be situated within a body of literature that focuses on determining the best plan of action for an autonomous vehicle, given a pedestrian behavior model. Less complex approaches use reactive control schemes ([40, 20]) that neither utilize a pedestrian model nor account for the delayed effects of the agent's current action. As a result, these approaches often lead to sub-optimal decisions. Another common approach is to use deterministic pedestrian behavior models to generate paths that avoid dynamic and static obstacles. The path can subsequently be executed using a feedback controller ([50, 28]). However, both approaches ignore the uncertainty in pedestrian intention estimation. Recent work has addressed this issue by formulating the problem as a POMDP, and then solving it by either using techniques from deep reinforcement learning (RL) [13, 41] or online POMDP solvers [52, 11, 29, 45].

The works most closely related to this work use online POMDP planning for the task of navigating under uncertainty[7, 47, 33, 44, 9, 22, 21]. Bai et. al [7] tackled the complex task of navigation among pedestrians using a two step process. The block diagram for their approach is presented in Fig. 2.1. They used hybrid A^* to obtain a sequence of steering angles that can guide

the vehicle to its goal, and then used a POMDP planner which reasoned over the uncertainty in nearby pedestrians' intention to control the speed over that path. This two step process can lead to undesirable stalling of the vehicle. Luo et. al [33] compared the two-step planner's performance against a dynamic hybrid A^* approach that planned over both heading angle and speed, and the dynamic hybrid A^* approach outperformed other planners in all of their evaluation metrics. However, it led to collisions with pedestrians because unlike POMDP planning, dynamic hybrid A^* path planning does not have the capability to handle pedestrian intention uncertainty. MAGIC [29] showed the effectiveness of using macro actions, a combination of both steering and speed in POMDP planning for autonomous driving in crowded environments. This suggests that the POMDP planner should control both degrees of freedom for safe and efficient planning. However, MAGIC introduces an additional step of learning macro actions which, we argue, is not necessary if suitable roll-out policies are used.

Liang et al. [30], Sathyamoorthy et al. [39] and Fan et al. [32] used PPO [41] to train an RL policy that directly maps sensor data to vehicle velocity for collision avoidance with dynamic obstacles. However, these RL agents are hard to train for long range navigation tasks in complex environments where reward is sparse [19]. SA-CADRL [12] uses a global planner [14] to generate way-points/sub-goals in close proximity, and used an RL planner to obtain socially acceptable collision free path between those way-points. To solve long range navigation tasks with just static obstacles, PRM-RL [19] uses motion planning techniques, primarily sampling based methods for generating a roadmap using the RL agent to determine connectivity, rather than the traditional collision free straight line interpolation in C-space. RL-RRT [16] applies similar idea but also imposed kinodynamic constraints on the local RL planner. They showed the effectiveness of offline methods in guiding the optimal decision search.

Our work shows that online POMDP planning over increased degrees of freedom is achievable and more effective than controlling only a subset, without the need to learn and incorporate macro actions [29]. The advantages of expanded space POMDP planning comes at the cost of higher computational complexity which can be offset by the use of offline methods. In practice, the

"offline" portion of the computation can be carried out online, but at a slower rate than the POMDP planning. The purpose of this work is to demonstrate that POMDP planning is an effective tool that is capable of rapidly solving large-horizon planning problems, provided it can be guided by effective roll-out policies. To the best of our knowledge, this is the first work that combines POMDP planning over multiple degrees of freedom with multi-query motion planning approaches for real time navigation in continuous dynamic environments with multi-modal process uncertainty.

Chapter 3

TECHNICAL APPROACH

This section describes the different technical components of our approach including our POMDP model, the DESPOT algorithm, and the multi-query planning techniques that underpin its performance.

3.1 POMDP Preliminaries

The Markov Decision Process (MDP) is a mathematical framework for representing a broad class of sequential decision making problems. A POMDP is a generalization of an MDP in which the agent cannot directly observe the underlying state. Instead, it must maintain a probability distribution over the set of possible states, based on a set of observations and observation probabilities, and the underlying MDP.

A POMDP is defined by a tuple $(S, A, Z, T, O, R, \gamma)$, where S is the state space, A is the action space, Z is the observation space, T is the transition model, O is the observation model, R is the reward model, and γ is the discount factor. When the system is in state $s \in S$ and takes an action $a \in A$, it reaches state $s' \in S$ with probability $T(s, a, s')$ and gets an observation $z \in Z$ with probability $O(s', a, z)$. The reward model R is specified by a function $R(s, a, s')$ which specifies the immediate reward of transitioning from state s via action a to state s'.

A policy for a POMDP is a function π that specifies the action $a = \pi(b)$ at any given belief over the state space b. Online POMDP solvers generate a policy that maximizes the expected total reward from the current belief b:

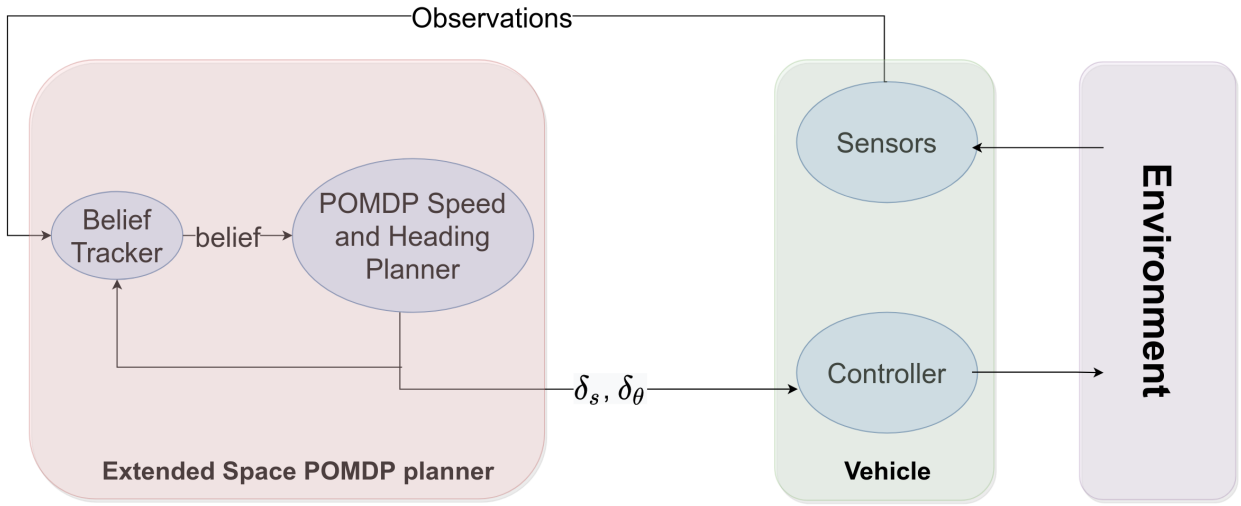

Figure 3.1: Extended Space POMDP based planning for autonomous driving.

$$V_\pi(b) = \mathrm{E}(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t)) | b_0 = b) \tag{3.1}$$

3.2 Problem formulation as a POMDP

Our approach utilizes a POMDP to model both the agent and the dynamic obstacles around it, generating control solutions that account for uncertainty in the environment. Fig. 3.1 represents the block diagram of the proposed approach.

3.2.1 State Modeling

The state vector in our dynamic environment navigation task POMDP consists of the vehicle state and a vector of dynamic obstacle states. The vehicle state consists of position (x_c, y_c), orientation θ_c, current speed v_c and its goal location g_c. The state vector contains n_{ped} pedestrian states whose future motion intentions are not directly observable, contributing uncertainty in the problem formulation. The state of the i^{th} pedestrian consists of its position (x_i, y_i), speed v_i, and its intended goal location g_i. The intention of a pedestrian is modeled as a goal location, which is hidden from the vehicle and must be inferred from its observed behavior.

3.2.2 Action Modeling

The action space in the navigation POMDP consists of a two dimensional vector where the agent chooses both steering (the change in the orientation angle, δ_θ) and velocity (the change in vehicle's speed, δ_s) controls, with the range of possible values for each being dependent on the vehicle state. Further details on this are available in Section 4.3. There is also a SUDDEN BRAKE (SB) action that immediately stops the vehicle to avoid collision with pedestrians in unexpected scenarios.

3.2.3 Observation Modeling

An observation in our POMDP model is a vector consisting of the vehicle position and the discretized position of all the n_{ped} pedestrians. Given state-of-the-art sensing technology and the effectiveness of filtering techniques, our model assumes no observation noise for these variables (empirically, small noise here does not materially affect agent policy). As a pedestrian's intention is the partially observable variable in our model, we have to infer it from the observations received over time, hedging against estimation uncertainty during decision making.

3.2.4 Reward Modeling

The POMDP's reward model guides the vehicle towards an optimal driving behavior which is safe, collision-free, and reaches the goal efficiently. We considered the following rewards in our model.

- Goal Reward: If the vehicle reaches within distance D_g to the goal, then there is a large positive reward R_{goal}. This reward is modeled to encourage the vehicle to reach its goal.
- Obstacle Collision Penalty: If the vehicle passes within a distance D_{obs} to the static obstacle, then there is a substantial negative reward of R_{obs}. This reward is modeled to prevent the vehicle from running into static obstacles.

- Pedestrian Collision Penalty: If the vehicle is moving and passes within a distance D_{ped} to a pedestrian, then there is a substantial negative reward of R_{ped}. If the vehicle is stationary, then we assume the pedestrian is responsible to avoid it. This reward is modeled to ensure safety of the pedestrians as well as the vehicle.

- Low Speed Penalty: If the vehicle is driving slower than it's maximum possible speed v_{max}, then there is a small negative reward $R_{speed} = (v_c - v_{max})/v_{max}$. This reward is modeled to encourage the vehicle to drive fast whenever possible.

- Sudden Stop Penalty: If the vehicle chooses the SB action, then there is a negative reward of R_{SB}. This reward is modeled to incentivize the policy against frequent "sudden brake" action, and exploring paths where that action can be avoided.

- There is also a small negative reward of R_t for every planning step. This reward is included to discourage longer paths.

3.2.5 Generative Model G

For many problems, it is difficult to explicitly represent the probability distributions T and Z. Some online POMDP solvers, however, only require samples from the state transitions and observations. As a consequence, it is beneficial to use a generative model which implicitly defines T and Z, even when they cannot be explicitly represented. G stochastically generates a new state, observation, and reward given the current state and action: $s', o, r = G(s, a)$. In our generative model, for a given POMDP state s and action a, we simulate the vehicle forward by applying a for time step Δt and move all pedestrians towards their sampled goal location. The i^{th} pedestrian is moved towards g_i by a distance of $v_i \Delta t + \omega_i$, where ω_i is a small random noise. While more complex pedestrian models exist (e.g. PORCA [33]), the choice of dynamic object model is regarded as an interchangeable component of the presented architecture and is not framed as a contribution of this work.

3.3 Solving POMDPs Online with DESPOT

For a given belief b_0, we do online POMDP planning to determine the best vehicle action at that belief. Online POMDP planning is performed by doing a tree search over the belief space starting from the belief node b_0. In the generated belief tree, each node represents a belief, and each edge represents an action-observation pair. The belief tree is searched using post-order traversal. At every leaf node, a default policy is simulated to obtain a lower bound on its value. At each internal node, the best action is chosen using Bellman's principle of optimality:

$$V(b) = \max_{a \in A} \{ \sum_{s \in S} b(s) R(s,a) + \gamma \sum_{z \in Z} p(z/b,a) V(\tau(b,a,z)) \} \tag{3.2}$$

which recursively computes the maximum value of action branches and the average value of observation branches. This results is an approximately optimal policy π for the current belief b_0. The vehicle can then execute the first action of the policy, $\pi(b_0)$. A complete belief tree grows on the order of $O(|A|^D|Z|^D)$ where D is the maximum depth of the belief tree. When the action or observation space is large, it becomes impractical to construct or search the full belief tree within the limited computational budget.

DESPOT [52], a state of the art anytime algorithm, whose key strengths include handling large observation spaces addresses this issue. Since the second sum in equation 3.2 computes an average value over observation branches, there is no need to examine all observation branches to estimate the average and identify an approximately optimal action. A sampled subset of observations branches may be sufficient to estimate this average. The key idea of DESPOT is to summarize the execution of all policies under K sampled scenarios. Under each scenario, a policy traces out a path in the belief tree (Fig. 3.2). This path corresponds to a particular sequence of action chosen by the policy and observation received. The belief tree generated using the DESPOT algorithm is a sparsely sampled belief tree which contains only the belief-tree nodes and edges traversed by all possible policies under the sampled scenarios. While the original belief tree contains $O(|A|^D|Z|^D)$ nodes, the DESPOT tree contains only $O(|A|^D K)$ nodes, leading to dramatic improvement in com-

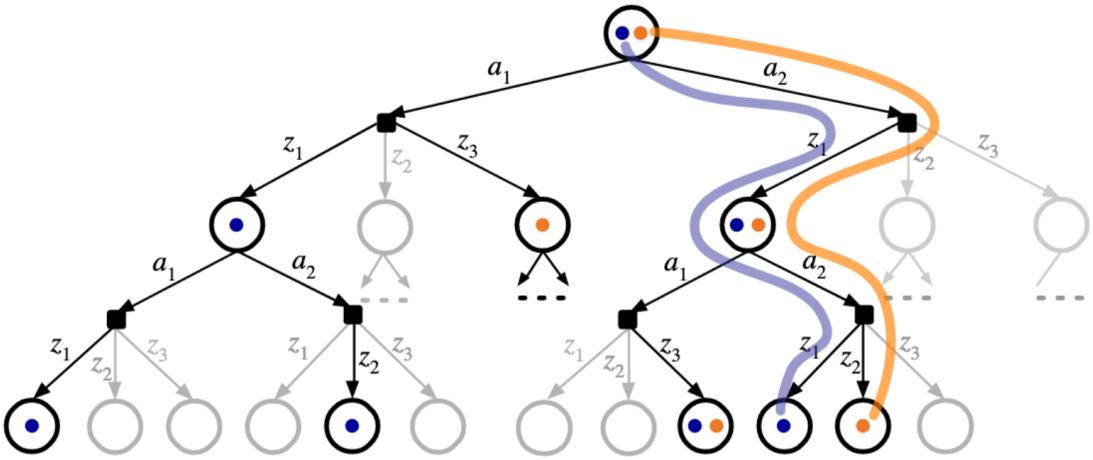

Figure 3.2: A belief tree of height H= 2(gray) and a corresponding DESPOT tree (black) obtained with 2 sampled scenarios, shown in blue and orange.The blue and orange curves indicate the execution paths of a same policy under the two scenarios. Image borrowed from [7].

putational efficiency for moderate K values. DESPOT builds its tree incrementally by performing a heuristic search guided by a lower bound and an upper bound on the value at each belief node in the tree.

We calculate the lower bound at a belief leaf node b_l by simulating a roll-out policy for all the scenarios at that belief. For LS, the roll-out policy executes the hybrid A^* path. For our proposed formulation (ES), the roll-out policy executes a path from the vehicle's current location to its goal aided by the use of a multi-query planner (e.g., PRM or FMM). We use a reactive controller to determine vehicle speed along the path. If there are no pedestrians within distance D_{far} from the vehicle, then it increases its speed by 1 m/s. If there are pedestrians within distance D_{near} to the vehicle, then it decreases its speed by 1 m/s. Otherwise it maintains its current speed. The roll-out policy is run for a fixed, predefined number of steps M or until the termination criteria has been met.

We calculate the upper bound at b_l by averaging the upper bound for all the scenarios at b_l. For a scenario, if the vehicle is not stationary and is within distance D_{ped} from any pedestrian, then the bound is R_{ped}. Otherwise, it is $\gamma^t R_{goal}$ where t is the time taken by the vehicle to reach

the goal along the chosen path assuming that the vehicle drives at its maximum speed with no dynamic obstacles (e.g., pedestrians) around.

DESPOT generates a policy tree from this information, with the controller selecting the action at the root of the tree with the greatest expected reward.

3.4 Fast Marching Method for Multi-Query Path Planning

The Fast Marching Method (FMM) is an algorithm for tracking and modeling the motion of a physical wave interface [36]. The interface is a flat curve in 2-D and a surface in 3-D or higher dimensions. It efficiently solves the Eikonal equation:

$$1 = F(x)|\nabla T(x)| \tag{3.3}$$

where x is the position, $F(x)(\geq 0)$ is the expansion speed of the wave at that position, and $T(x)$ is the time taken by the wave interface to reach x from its source.

Given the wave's source point and the expansion speed, F defined over all points in the environment, FMM calculates the time T that the wave takes to reach those points. Since $F > 0$, the wave can only expand and it can be shown that the T(x) function (originated by a wave that grows from one single point) has only one global minima at the source and no local minima. This method is effective in obtaining a path from any given point in the environment to the wave's source point using gradient descent [49].

Assume the environment displayed in Fig. 3.3 where black shapes represent static obstacles in the environment and the symbol G in green denotes the goal location of the vehicle. In order to obtain a path from any given point in the environment to the goal location, we need to solve the Eikonal equation. To solve equation 3.3 via FMM, we discretize the environment into grid cells, assigning $F = 0$ for those grid cells where static obstacles are present and $F = 1$ everywhere else in the environment. We let the wave originate from the vehicle's goal location and then solve equation 3.3 using the discrete solution that Sethian proposed in [42] to get a grid map of T values for all the cells. The heat map of the time values is displayed in Fig. 3.4 with the blue color denoting low value and the red color denoting high value. We then apply the Sobel operator in a 3×3 neighborhood of every grid cell on the grid map of time values to obtain the direction of the gradient at that cell. Let the symbol S in green denote the position of the vehicle in the environment (Fig.3.5). We use the position coordinates of the vehicle to determine the cell of the environment in which the vehicle is present. In order to find a path from that cell to the vehicle's goal location for the roll-out policy,

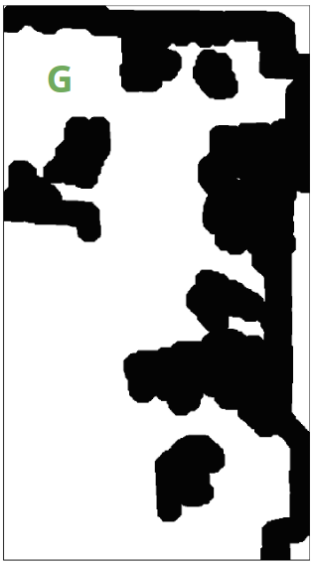

Figure 3.3: Environment with static obstacles (white region denotes free space and black region denotes occupied space. G denotes the goal location of the vehicle. Image borrowed from [1]

we keep moving some predetermined and fixed α units in the opposite direction of the gradient at that cell until the goal is reached. The obtained path from S to G can be seen in Fig.3.6.

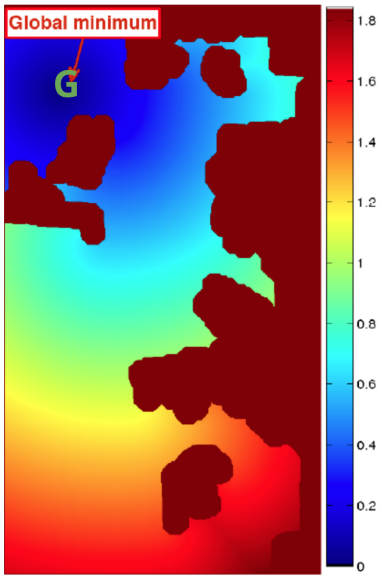

Figure 3.4: Heat map of the time values obtained by solving the Eikonal equation using FMM where G is the starting point of the wave. Image borrowed from [1]

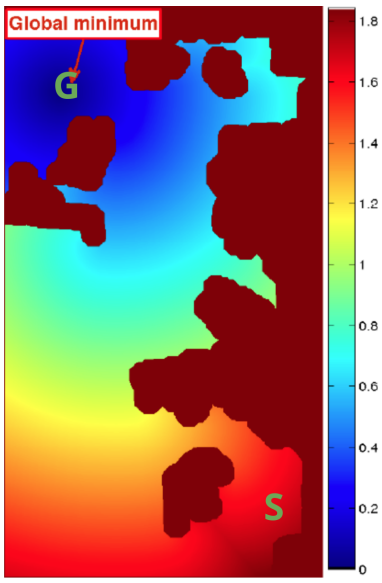

Figure 3.5: S denotes the position of the vehicle in the environment. Image borrowed from [1]

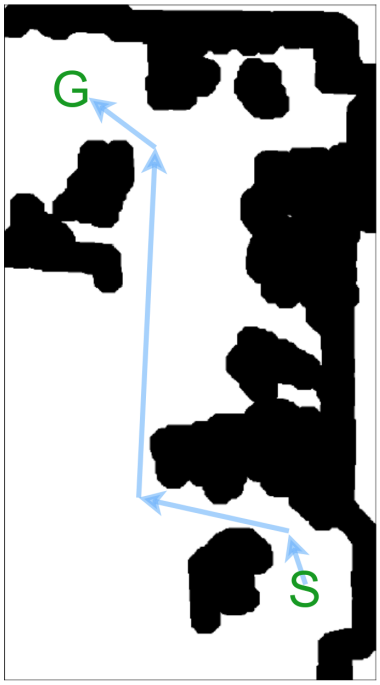

Figure 3.6: Path obtained from S to G by moving α units in the opposite direction of the gradient until the goal location G is reached. Image borrowed from [1]

3.5 Probabilistic Roadmaps for Multi-Query Path Planning

The Probabilistic Roadmap (PRM) is a well known method for path planning in high dimensions for robots in static environments. The method constructs a graph whose nodes correspond to collision-free configurations in the space and whose edges correspond to feasible paths between these configurations [24]. This method can be used for any type of holonomic robot. For a holonomic vehicle moving on the 2-D plane, the graph nodes correspond to (x, y) coordinates in the environment and the edges correspond to collision free linear paths between those points.

Assume the environment displayed in Fig. 3.7 where black shapes represent static obstacles in the environment and the vehicle's goal location is near the top right corner. In this work, we assigned vehicle's goal location as a node in the PRM, and randomly sampled $N_{prm} - 1$ more collision-free configurations or nodes in the environment (Fig. 3.8). An edge can be added between two nodes in the generated PRM if there is a straight line collision-free path between those two nodes. We added edges by connecting each node to its k nearest neighbors to which a collision-free straight line path is possible (Fig. 3.9). The euclidean distance between the two nodes represent the weight of the edge between them. We can then use any graph search algorithm to find the shortest path from any node to the the node in the PRM which corresponds to the vehicle's goal location. This was repeated for all the nodes in the PRM to find the shortest path and the cost of that path. Using this PRM and the pre-computed shortest paths, we can find a path from any point in the environment to the vehicle's goal location for the roll-out policy. For any given position of the vehicle in the environment, we consider only the PRM nodes within a predetermined fixed distance, d_{PRM} from the vehicle to find the path (Fig. 3.10). For all the nodes within distance d_{PRM} from the vehicle to which straight line traversal is possible, we compute the euclidean distance to reach those nodes. Since we have already computed the path from every PRM node to the goal location, we use that along with the calculated euclidean distance to calculate the total cost to reach the goal location from the vehicle's location via that PRM node (Fig. 3.11). The path with the least total cost is chosen for the roll-out policy (Fig. 3.12).

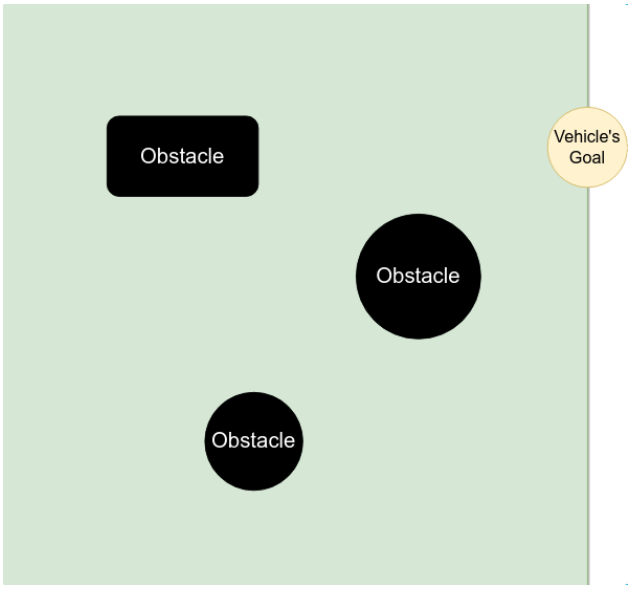

Figure 3.7: Environment with static obstacles

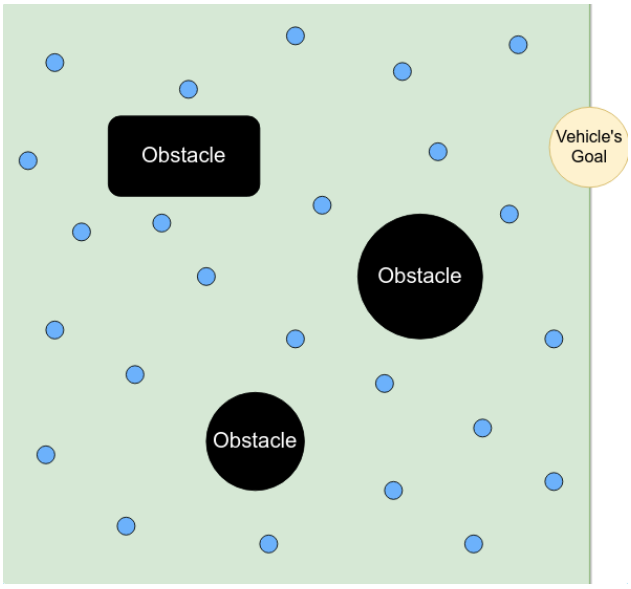

Figure 3.8: Sampling collision-free configurations (blue dots) in the environment to generate a PRM

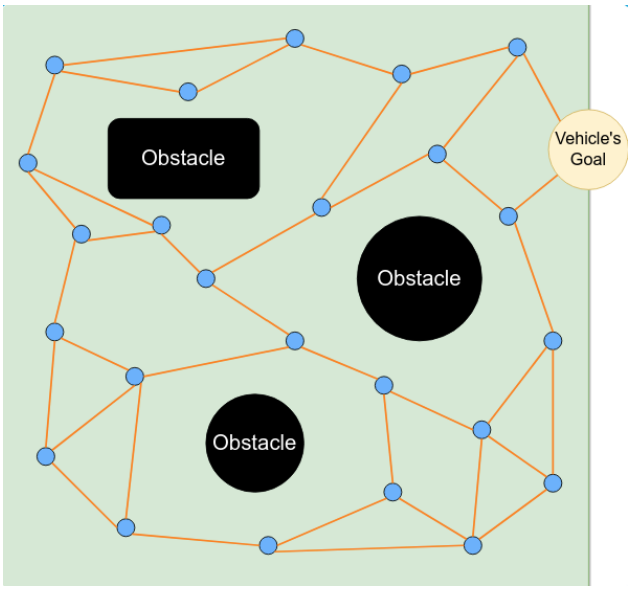

Figure 3.9: Connecting every node to its k nearest neighbors in the PRM

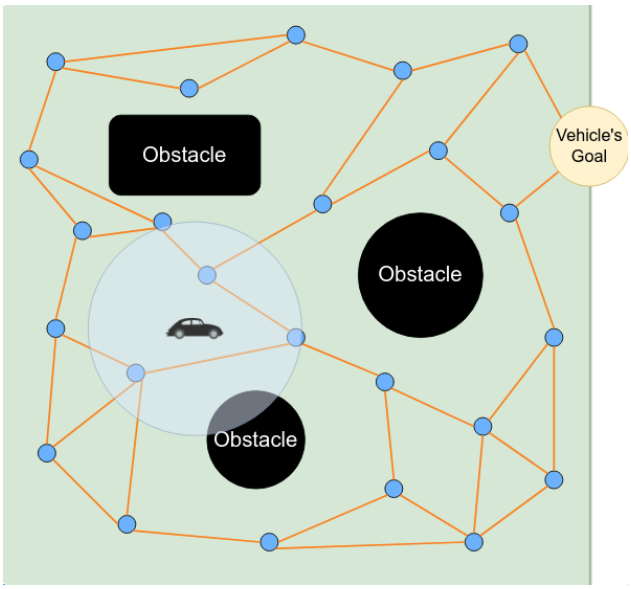

Figure 3.10: Vehicle and the PRM nodes within distance d_{PRM} from the vehicle in the PRM

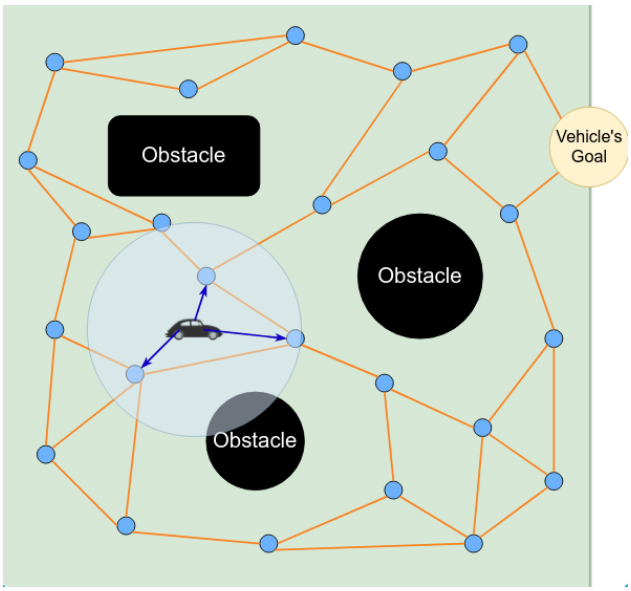

Figure 3.11: Calculating total cost from the vehicle to its goal location for all the possible paths

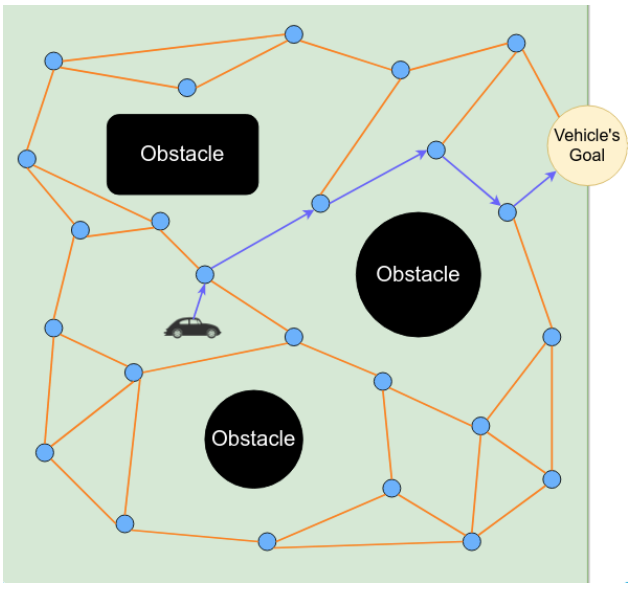

Figure 3.12: Lowest cost path from vehicle to its goal location using the sampled PRM

3.6 Tracking POMDP Belief

The partially observable variables in the POMDP formulation are pedestrian intentions (goal locations) that are inferred by the belief tracker based on the series of observations received. Since in practice there tends to be a finite number of pedestrian goal locations for a given environment, the belief over all such intentions for each pedestrian forms a discrete probability distribution. Changes in goal can also be captured by the POMDP's belief tracker. For each pedestrian being attended to, the belief tracker observes their movement from (x, y) to (x', y'), calculates their velocity v, and updates the belief $b(g)$ over all the possible intentions to $b'(g)$ using the following update formula: $b'(g) = \eta p(x', y'|x, y, v, g, M)b(g)$, where η is a normalization constant. Based on the chosen pedestrian model M, $p(x', y'|x, y, v, g)$ will be directly proportional to the progress the pedestrian made towards goal g.

Chapter 4

EXPERIMENTS

In this section we explain our simulation environment and different experimental scenarios. We also provide specific details about different planners and parameter values used for the experiments. The open source code for the experiments is hosted at https://github.com/himanshugupta1009/extended_space_navigation_pomdp.

4.1 Simulation Environment

The environment in our simulator is a 100 m $\times$ 100 m square field. It is assumed that the four corners of the square are the possible goal locations for the pedestrians. The autonomous vehicle is modeled as a holonomic vehicle whose starting position is in the bottom left half and goal location is in the top right half of the field as can be seen in Fig. 4.1. Using the Kinova MOVO robot as a representative example of this class of platform, we set the vehicle's maximum speed to 2 m/s. Pedestrians are assigned one of the four possible goal locations at random. As soon as a pedestrian reaches its goal location, it is removed from the environment and a new pedestrian is spawned randomly along one of the edges of the field. Its goal location is chosen from the two goals on the opposite edge at random. This is done to ensure that there are a fixed number of moving pedestrians in the environment at any moment of time. The pedestrian simulation model is same as the model used by the POMDP generative function in Section 3.2.5. However, since it is merely a component of our simulator, it can be replaced by an alternative pedestrian model (e.g. PORCA [33]) without much effect on the performance of the proposed approach so long as

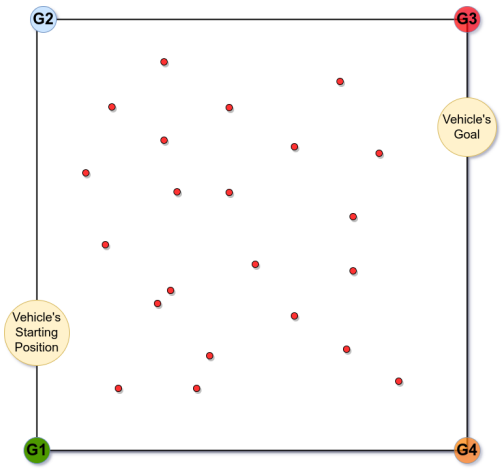

Figure 4.1: Simulation Environment

it is congruent with pedestrian behavior in the environment. This simulator was built using the high-level, high-performance programming language Julia [8].

4.2 Experimental Scenarios

We have designed three different scenarios to compare our planner's performance against a widely adopted baseline method (LS planner) [7]. They are as follows:

4.2.1 **Scenario** 1

There are no static obstacles in the environment. It resembles an open field. We designed this to analyze the proposed planner's performance when there is plenty of empty space for the vehicle to explore to avoid collision with pedestrians.

4.2.2 **Scenario** 2

There are six small static circular obstacles that are scattered throughout the environment. It resembles a cafeteria setting. We designed this to analyze the proposed planner's performance when there is less empty space available but it is distributed throughout the field.

4.2.3 **Scenario** 3

There is a large static circular obstacle in the bottom right corner of the environment. It resembles a L shaped lobby. We designed this to analyze the proposed planner's performance when the empty space is available only in certain parts of the environment which forces the vehicle to navigate among pedestrians in a limited space.

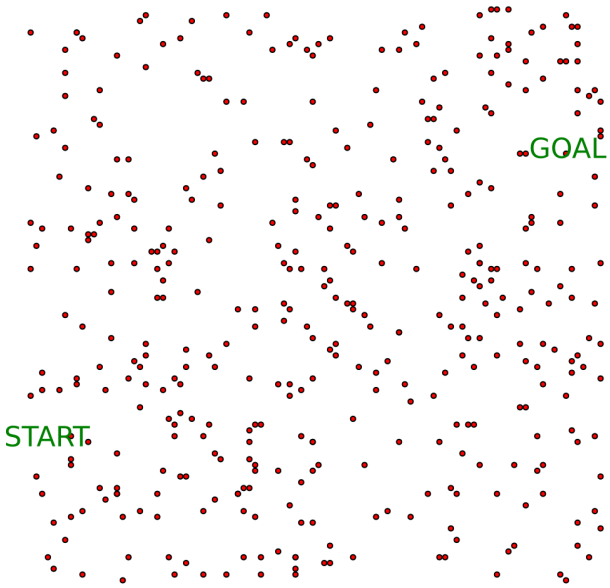

Figure 4.2: Scenario 1: Open field

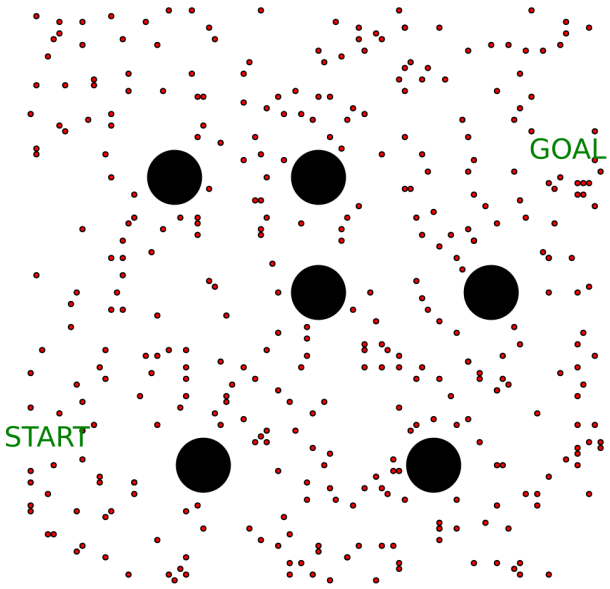

Figure 4.3: Scenario 2: Cafeteria setting

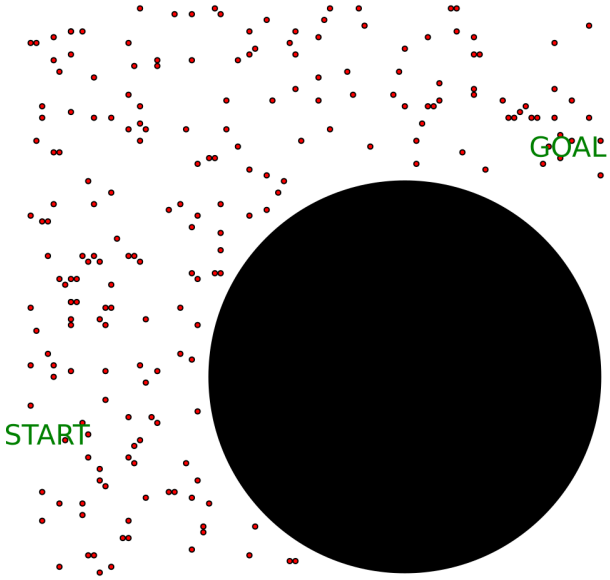

Figure 4.4: Scenario 3: L shaped lobby

4.3 Planners

The alternative experimental planners tested in the experiments are described below. All planners use the POMDPs.jl [18] implementation of DESPOT from the ARDESPOT.jl package.[1]

4.3.1 *LS* planner

This is the baseline approach, $1D$-A^* against which we have compared our proposed planner's performance. At every time step, the hybrid A^* algorithm finds a path from vehicle's current position to its goal. The path generated by Hybrid A^* on this landscape is then used in conjunction with a POMDP solver that determines the optimal speed given the fixed path and pedestrians located around the vehicle.

The Hybrid A^* algorithm is an extension to A^* that can generate a drivable path for the vehicle over continuous state space, and notably was used for autonomous mobile robot path planning during the DARPA Urban Challenge [48]. The path from vehicle's current position to its goal location is denoted by ρ, and is defined as a sequence of points $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$ equally spaced along ρ. The path cost is defined as

$$C(\rho) = \sum_{i=0}^{n} (\lambda^i C_{st}(x_i, y_i) + \lambda^i C_{ped}(x_i, y_i))$$

Hybrid A^* finds a minimum cost path from the vehicle's current position to its goal location. The path cost is the sum of two components, 1) C_{st} to penalize collisions with static obstacles and 2) C_{ped} to penalize collisions with pedestrians. We reduce the path cost exponentially over time by a fixed discount factor $\lambda \in (0, 1]$ to more heavily consider cost estimates at the beginning of the path due to increasing uncertainty [7]. In this approach, every pedestrian is modeled as a static obstacle at the center of a potential field with size proportional to the uncertainty around their intended goal. When pedestrian's intention is highly uncertain, we place a large potential field around the pedestrian's current location. Otherwise, we place a potential field around the

[1] https://github.com/JuliaPOMDP/ARDESPOT.jl

pedestrian's most likely path [7]. The path planner has 36 search actions from $-170°$ to $180°$ at $10°$ intervals.

The corresponding POMDP's reward model is the same as that described in Section 3.2.4 with the exception of the obstacle collision penalty, which is omitted since collision with static obstacles is avoided by the path planner. LS uses DESPOT to determine the best possible δ_s out of $\{-1m/s, 0m/s, 1m/s, SB\}$ at every time step along the generated path.

4.3.2 ES planner

We propose two ES planners, $2D\text{-}FMM$ and $2D\text{-}PRM$. At every time step, ES selects both, δ_θ and δ_s for the vehicle. Since DESPOT does not perform well for continuous or large action space problems, we choose a small discrete set of actions depending on the vehicle's state variables (x_c, y_c, v_c). When $v_c = 0$, there are 9 possible actions. It can either stay stationary (i.e $\delta_s = 0$ m/s) or increase its speed (i.e $\delta_s = 1$ m/s) while choosing a δ_θ. There are 7 possible choices for δ_θ from $-45°$ to $45°$ at $15°$ intervals. We add another potential value for δ_θ related to potential roll-out policies called δ_{RO}, which changes the vehicle's orientation according to the FMM or PRM roll-out policy at (x_c, y_c). If $v_c \neq 0$, then there are 11 possible actions. The planner can choose to either increase ($\delta_s = 1$ m/s) or decrease ($\delta_s =$ -1 m/s) its speed without changing its orientation (i.e. $\delta_\theta = 0°$), or maintain its current speed (i.e $\delta_s = 0$ m/s) and select from 8 possible δ_θ choices mentioned above, or apply the SB action.

Depending on the planner, DESPOT uses either FMM (Section 3.4) or PRM (Section 3.5) to obtain a path for the scenarios at a belief node. To evaluate the lower bound at that belief node, the roll-out policy executes a reactive controller to determine speed over that path.

4.4 Experimental Details

For each scenario, we ran sets of 100 different experiments with different pedestrian density in the environment. The number of pedestrians in the environment varied from 100 to 400 (in increments of 100). In each experiment, pedestrians were assigned random starting points and intentions. The performance of different planners was compared for that sampled environment under the same random seed for noise in simulated pedestrian motion. In simulations, the planning time for each step is 0.5 seconds. For $1D$-A^*, we devote 0.15 seconds for path planning, and 0.35 seconds for speed planning by solving the corresponding POMDP. For $2D$-FMM and $2D$-PRM, all of the planning time is devoted to solving the POMDP because the multi-query motion planning needs to be computed only once for the environment. The online POMDP solver reasons over the uncertainty in intentions of 6 nearest pedestrians (i.e. $n_{ped} = 6$). DESPOT performs online tree search with 100 sampled scenarios. We define a trajectory to be unsafe if at any time step the moving vehicle gets within 1 m distance of a pedestrian.

Chapter 5

RESULTS and DISCUSSION

Table 5.1: Holonomic Vehicle Planner Performance Comparison.

Scenario	1D-A*		2D-FMM			2D-PRM		
(# Ped)	Time (in s)	# SB action	Time (in s)	# Outperformed	# SB action	Time (in s)	# Outperformed	# SB action
1 (100)	77.02 ± 0.69	0.34 ± 0.06	**64.04 ± 0.37**	99	**0.21 ± 0.04**	**64.06 ± 0.36**	99	**0.16 ± 0.04**
1 (200)	87.73 ± 0.86	**0.47 ± 0.06**	**69.23 ± 0.54**	98	**0.44 ± 0.06**	**69.16 ± 0.64**	98	**0.36 ± 0.06**
1 (300)	99.14 ± 1.03	**0.49 ± 0.06**	**76.93 ± 0.76**	100	0.94 ± 0.08	**76.5 ± 0.69**	99	0.78 ± 0.08
1 (400)	115.85 ± 1.81	**0.94 ± 0.10**	**90.03 ± 1.42**	94	1.42 ± 0.12	**91.31 ± 1.09**	93	1.25 ± 0.11
2 (100)	78.97 ± 0.86	0.31 ± 0.05	**66.05 ± 0.44**	95	0.24 ± 0.04	**65.92 ± 0.42**	93	**0.16 ± 0.03**
2 (200)	89.44 ± 0.85	**0.44 ± 0.06**	**72.24 ± 0.69**	95	**0.48 ± 0.06**	**72.21 ± 0.75**	98	**0.47 ± 0.06**
2 (300)	102.67 ± 1.31	**0.5 ± 0.07**	**81.95 ± 0.95**	98	0.94 ± 0.09	**83.74 ± 1.07**	93	1.05 ± 0.10
2 (400)	114.46 ± 1.94	**0.56 ± 0.06**	**90.65 ± 0.98**	93	1.72 ± 0.11	**92.78 ± 1.49**	91	1.64 ± 0.11
3 (100)	82.07 ± 0.81	0.41 ± 0.06	**73.18 ± 0.64**	93	**0.29 ± 0.04**	**72.27 ± 0.66**	92	**0.22 ± 0.04**
3 (200)	94.87 ± 1.03	**0.35 ± 0.06**	**79.53 ± 0.64**	93	0.56 ± 0.06	**79.23 ± 0.76**	91	0.49 ± 0.06
3 (300)	107.26 ± 1.35	**0.46 ± 0.07**	**87.89 ± 0.93**	97	0.98 ± 0.08	**87.41 ± 1.12**	95	1.0 ± 0.10
3 (400)	122.73 ± 2.32	**0.76 ± 0.09**	**95.92 ± 1.02**	97	1.43 ± 0.11	**95.87 ± 1.20**	95	1.39 ± 0.10

The average travel time, the number of trajectories in which each proposed planner outperformed the baseline in terms of travel time, and the average number of SB actions for each algorithm over 100 trials. The standard error of the mean is indicated for averaged quantities. The best travel time and the least amount of SB action in each row are bolded. Multiple entries are bolded if they are statistically similar.

The results from our experiments are summarized in Tables 5.1 and 5.2. We computed the average travel time, the number of trajectories in which the proposed planner outperformed the baseline in terms of travel time, and the average number of times SB action was executed across 100 experiments for all the planners in different settings. Since all the planners have the SB action in their action space, they executed safe trajectories in every experiment. All of the experiments fulfilled the success criteria.

Experimental results for a holonomic vehicle are compiled in Table 5.1. The first key observation is that our proposed extended space planners, $2D\text{-}FMM$ and $2D\text{-}PRM$ executed paths that

took less travel time than the baseline without compromising safety. For each experimental setting, the best travel time is marked in bold in Table 5.1. The travel time for $2D\text{-}FMM$ and $2D\text{-}PRM$ is comparable across all the different settings which indicates that the sensitivity of travel time to the choice of motion planning algorithm used for generating effective roll-out policies is low. The differences in travel time for ES planners as compared to the LS planner across different settings are characterized in Fig. 5.16.

The baseline approach took more travel time on average primarily due to the segregation of planning problem in two components. The hybrid A^* algorithm generates a path without considering the vehicle's speed and by using ad-hoc techniques to handle uncertainty in pedestrian intention as described in Section 4.3.1. In most of the cases, the POMDP speed planner (LS) realizes that traveling along this fixed path at the vehicle's current speed can lead to a collision. As a result, it decides to either slow down or stop which increases the travel time. The decoupling of heading angle and speed forces the baseline approach to reason over uncertainty along just one path and the vehicle fails to perform efficient motion between moving pedestrians. On the other hand, the ES planners reason over uncertainty along multiple paths (Fig. 1.3) and often manage to find a path where they do not have to slow down or stop. Moreover, it is possible that the hybrid A^* path might not be obtained at every time step within the limited computation time. As a result, the system has to estimate the speed over the old path which was constructed considering the position of pedestrians and belief over their intention at the previous time step. This leads to sub-optimal decision making.

Another important observation is that both proposed planners outperformed the baseline approach in the metric of travel time at least 91% of the times across all the different settings. In densely crowded environments, moving to empty spaces nearer to the agent's goal (instead of staying idle and letting pedestrians pass) is an intuitively good strategy. This behavior is visible from the trajectories executed by $2D\text{-}PRM$ and $2D\text{-}FMM$ (Fig. 5.4, Fig. 5.8 and Fig. 5.12). They cover a wider area of the environment than $1D\text{-}A^*$. However, doing so can sometimes also result in the vehicle momentarily getting stuck behind a group of pedestrians that it did not reason

Figure 5.1: 1D-A*

Figure 5.2: 2D-FMM

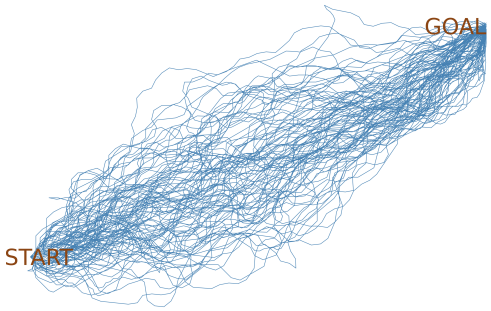

Figure 5.3: 2D-PRM

Figure 5.4: Trajectories executed by the holonomic vehicle using different planners across all 100 experiments in Scenario 1 with 400 pedestrians in the environment.

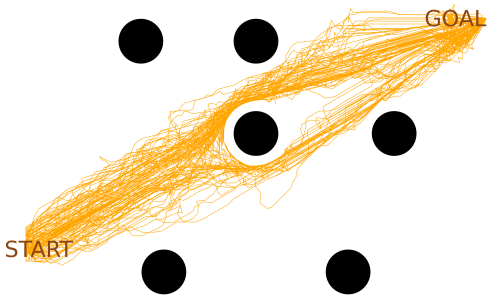

Figure 5.5: 1D-A*

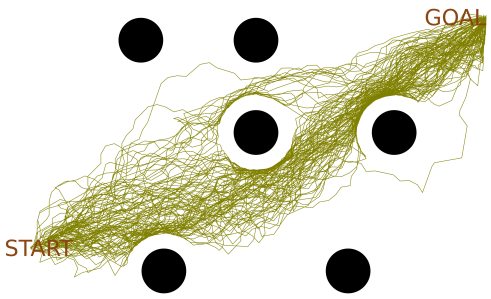

Figure 5.6: 2D-FMM

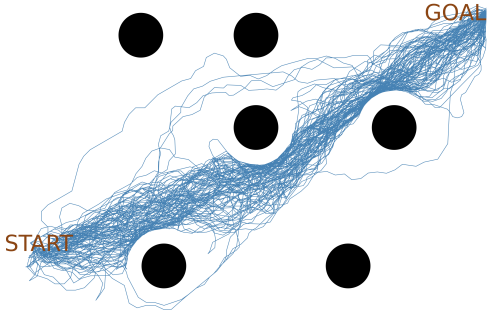

Figure 5.7: 2D-PRM

Figure 5.8: Trajectories executed by the holonomic vehicle using different planners across all 100 experiments in Scenario 2 with 400 pedestrians in the environment.

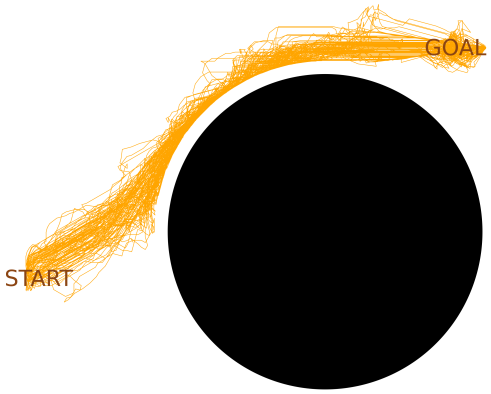

Figure 5.9: 1D-A*

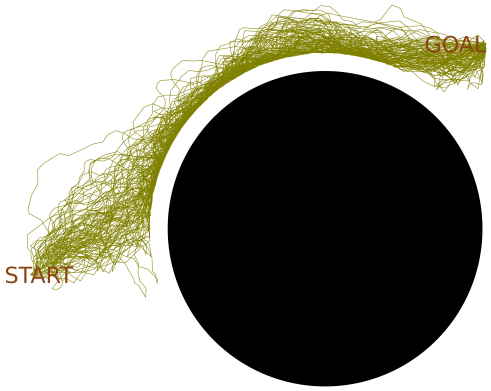

Figure 5.10: 2D-FMM

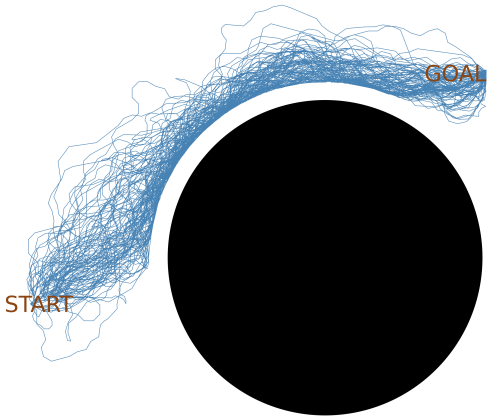

Figure 5.11: 2D-PRM

Figure 5.12: Trajectories executed by the holonomic vehicle using different planners across all 100 experiments in Scenario 3 with 400 pedestrians in the environment.

over earlier. This happened in the few experiments where the proposed planners took more travel time.

The extended space planners executed more SB actions on average than the baseline for settings with high pedestrian density under all the scenarios. This is mainly due to the availability of more choices of δ_θ for LS (36 choices) than ES (8 choices). The result with the least amount of SB action is marked in bold for every setting in Table 5.1. For low pedestrian density, ES planners took less or almost the same amount of SB actions as the LS planner. Densely populated environments have less free space for the vehicle to navigate. In that situation, having more choices for δ_θ allows the LS planner to find a path to move to open spaces that avoids collision with any pedestrian. On the contrary, due to limited choices, the ES planner decides to execute the SB action under the same situation. This is not a limitation directly due to extending the action space; rather it is a limitation of the particular online tree search algorithm, DESPOT, which does not work well for large or continuous action space problems. This demonstrates a need for online POMDP solvers that can better solve large or continuous action space POMDPs which is an active area of research [31].

We also performed experiments for a non-holonomic vehicle and compared the performance of LS and ES planners under Scenario 1. The vehicle is modeled as a Dubin's car with a max speed of 4 m/s. For LS, the hybrid A^* path planner has 19 search actions from $-45°$ to $45°$ at $5°$ intervals, and the POMDP model is same as that for a holonomic vehicle. For ES, an effective roll-out policy in the absence of static obstacles is to apply a steering angle β that modifies the vehicle's heading angle to follow a straight line path to the goal (subject to steering angle constraints). δ_{RO} can be calculated from β using the vehicle dynamics. The straight line roll-out is not effective in scenarios 2 and 3 as it could lead to collisions with static obstacles. Under those scenarios, a possible effective roll-out policy for a non-holonomic vehicle would be to execute a reactive controller over the path generated using the Fast Marching Square method [6].

The experimental results with the non-holonomic vehicle are summarized in Table 5.2. The ES planner ($2D$-NHV) took less travel time on average than the LS planner ($1D$-A^*) across each

Table 5.2: Non-Holonomic Vehicle Planner Performance Comparison.

	$1D\text{-}A^*$		$2D\text{-}NHV$		
#Ped	Time (in s)	# SB action	Time (in s)	# Outperformed	# SB action
100	53.62 ± 0.91	1.84 ± 0.11	**36.54 ± 0.45**	98	**0.26 ± 0.04**
200	72.54 ± 1.09	2.84 ± 0.14	**43.79 ± 0.85**	96	**1.04 ± 0.09**
300	95.43 ± 1.74	3.62 ± 0.15	**62.37 ± 1.57**	95	**2.55 ± 0.12**
400	110.41 ± 2.32	3.98 ± 0.15	**81.07 ± 1.80**	95	**3.54 ± 0.13**

This table shows the average travel time, the number of trajectories in which the proposed planner outperformed the baseline in terms of travel time, and the average number of SB actions over 100 trials under Scenario 1. The standard error of the mean is indicated for averaged quantities. The best travel time and the least amount of SB action in each row are bolded.

of the different settings, outperforming the baseline approach in this metric in at least 95% of the simulations. The sub-optimality in decision making due to the decoupling of heading and speed becomes more significant when the vehicle can travel at higher speeds. This is visible from the larger reduction in the travel time ratio for the non-holonomic vehicle (max speed = 4 m/s) in comparison to the holonomic vehicle (max speed = 2 m/s) across the different pedestrian densities (Fig. 5.16). Since the number of choices of δ_θ for LS planner (19 choices) is not significantly more than the number of choices for the ES planner (8 choices) for the non-holonomic vehicle, and due to decoupling, the LS planner executes more SB actions than the ES planner on average across all settings. The ES planner plans effectively and finds paths by covering a much wider space of the environment without having to suddenly stop (Fig. 5.15).

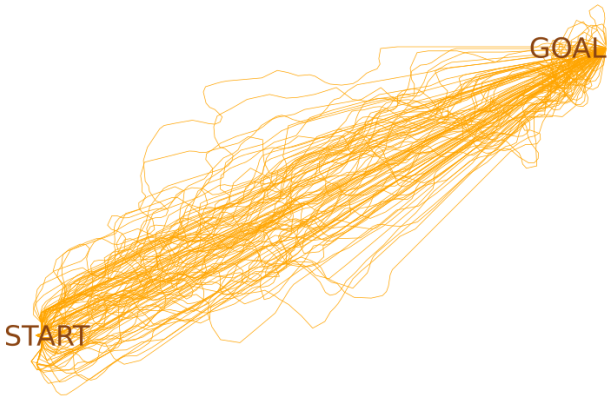

Figure 5.13: 1D-A*

Figure 5.14: 2D-NHV

Figure 5.15: Trajectories executed by the non-holonomic vehicle using different planners across all 100 experiments in Scenario 1 with 400 pedestrians in the environment.

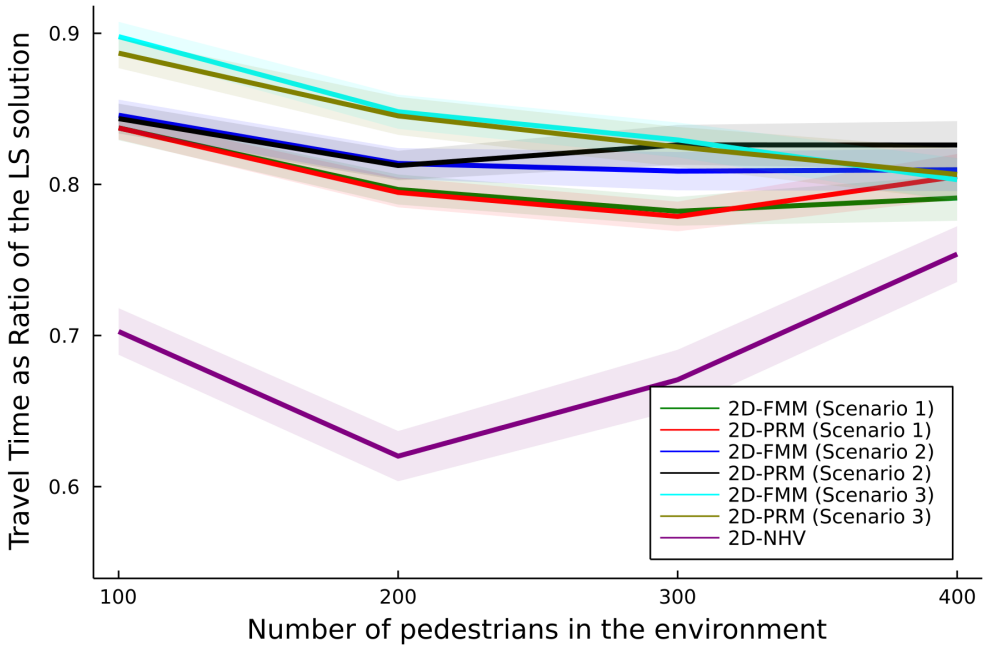

Figure 5.16: ES planners consistently outperform LS planners across each scenario, whether for holonomic or non-holonomic agents.

Chapter 6

CONCLUSION and FUTURE WORK

This work presents an intention-aware navigation system that uses an extended-space POMDP planner to generate efficient navigation policies in the presence of uncertainty introduced by other agents in the environment. In particular, for an autonomous vehicle navigating among pedestrians, solving a POMDP that controls all degrees of freedom(i.e. speed and heading), instead of a select few (i.e. speed over the Hybrid A^* path) results in faster trajectories. This additional control parameter enlarges the reachable state space and raises the need for practical roll-out policies from these states to find the sparse positive reward during the online tree search. Our system uses multi-query motion planning techniques like Fast Marching Methods or Probabilistic Roadmaps to efficiently generate effective roll-out policies. Our results show that the proposed extended space POMDP planners enable effective and safe autonomous driving in complex crowded environments. They also indicated that the choice of the motion planning technique for offloading the task of generating effective roll-out policies does not affect the planner's performance significantly.

There are multiple directions to work on in the future. We plan to implement the proposed extended space approach for non-holonomic vehicles in environments with static obstacles. Using a reactive controller on a straight-line path to the goal doesn't act as an effective roll-out policy for non-holonomic vehicles in the presence of static obstacles since this path could lead to collisions with static obstacles in the environment. The path obtained using PRM and FMM path may not be executable for a non-holonomic vehicle due to kinematic constraints. A possible solution to this issue is to use a Hamilton-Jacobi formulation which allows the generation of optimal trajectories

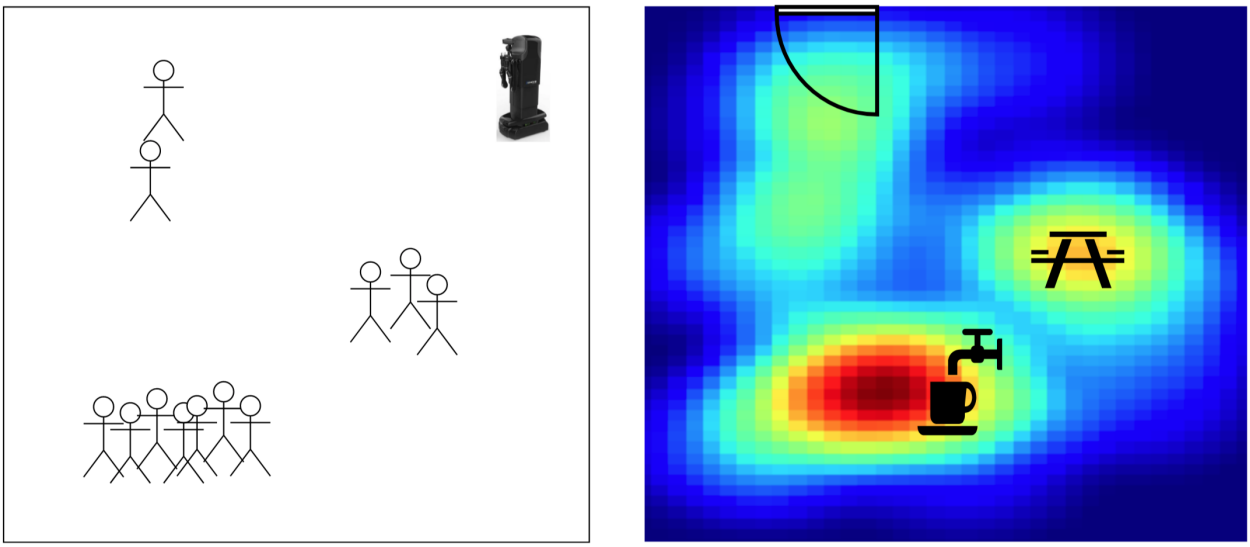

Figure 6.1: Goal-object association using heat maps and vision data

from any given point in the environment to the goal location while satisfying kinematic constraints on the vehicle.

In this work and prior related work, it was assumed that the possible human goal locations are known beforehand. However, this assumption will not be valid for most of the environments in real life. Consequently, an autonomous agent must have the capability to identify possible human goal locations in the environment. This agent can begin by observing and recording human trajectories in the environment to generate a heat map of the time humans spent in different parts of the room and to identify hotspots. It can then associate objects near these hotspots as potential goal locations, e.g. a coffee machine, a door or a table (Fig. 6.1). When this agent is interacting in a new environment, it can look for these objects to identify possible human goal locations.

Another possible extension is to use the proposed approach for planning in high dimensional space, for e.g., a robotic manipulator. Prior work has used the POMDP formulation for problems that involve human robot collaboration [38, 23]. Pellegrinelli et. al [38] used a POMDP framework to reason over uncertainty in human's intention while figuring out the best possible action for a manipulator in a table clearing task (Fig. 6.2). This POMDP was solved using hindsight optimization and requires offline computation of state-action value function for the manipulator. Since a

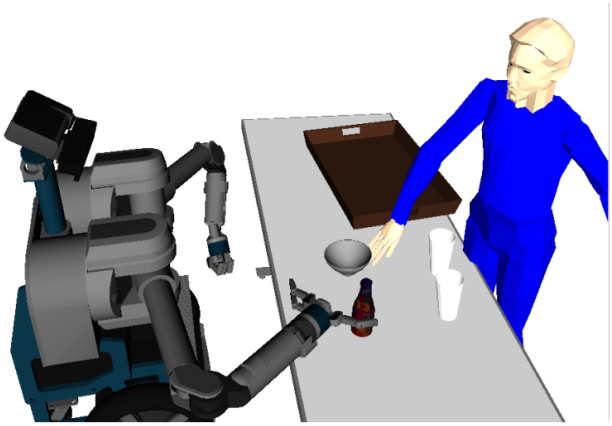

Figure 6.2: An example of human robot collaboration for clearing a table. Image borrowed from [38].

PRM is often used for motion planning of manipulators, our approach can use that sampled PRM for both choosing an action in the environment and for executing an effective roll-out policy during tree search, and can generate efficient motion for the manipulator.

Our work also demonstrated the need for online algorithms that can effectively solve large or continuous action space POMDPs. This is an active area of research and is something we are interested in working on.

Bibliography

[1] Fast Marching Method and Fast Marching Square DATE: 18th April 2022 URL: https://jvgomez.github.io/pages/fast-marching-method-and-fast-marching-square.html.

[2] The 6 Levels of Autonomous Driving Explained DATE: 18th April 2022 URL: https://www.autoweek.com/news/technology/a35492454/levels-of-autonomous-driving-explained/.

[3] Worker at VW factory crushed to death by robotic arm DATE: 18th April 2022 URL: https://performancedrive.com.au/worker-at-vw-factory-crushed-to-death-by-robotic-arm-0217/.

[4] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. International Journal of Robotics Research, 33(2):268–304, 2014.

[5] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 961–971, 2016.

[6] César Arismendi, David Álvarez, Santiago Garrido, and Luis Moreno. Nonholonomic motion planning using the fast marching square method. International Journal of Advanced Robotic Systems, 12(5):56, 2015.

[7] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In 2015 ieee international conference on robotics and automation (icra), pages 454–460. IEEE, 2015.

[8] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. SIAM review, 59(1):65–98, 2017.

[9] Maxime Bouton, Akansel Cosgun, and Mykel J Kochenderfer. Belief state planning for autonomously navigating urban intersections. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 825–830. IEEE, 2017.

[10] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In IEEE International Conference on Robotics and Automation (ICRA), pages 723–730. IEEE, 2011.

[11] Panpan Cai, Yuanfu Luo, David Hsu, and Wee Sun Lee. Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty. The International Journal of Robotics Research, 40(2-3):558–573, 2021.

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

[13] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In 2017 IEEE international conference on robotics and automation (ICRA), pages 285–292. IEEE, 2017.

[14] Yu Fan Chen, Shih-Yuan Liu, Miao Liu, Justin Miller, and Jonathan P How. Motion planning with diffusion maps. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1423–1430. IEEE, 2016.

[15] Yufan Chen, Miao Liu, Shih-Yuan Liu, Justin Miller, and Jonathan P How. Predictive modeling of pedestrian motion patterns with bayesian nonparametrics. In AIAA guidance, navigation, and control conference, page 1861, 2016.

[16] Hao-Tien Lewis Chiang, Jasmine Hsu, Marek Fiser, Lydia Tapia, and Aleksandra Faust. Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies. IEEE Robotics and Automation Letters, 4(4):4298–4305, 2019.

[17] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. Ann Arbor, 1001(48105):18–80, 2008.

[18] Maxim Egorov, Zachary N. Sunberg, Edward Balaban, Tim A. Wheeler, Jayesh K. Gupta, and Mykel J. Kochenderfer. POMDPs.jl: A framework for sequential decision making under uncertainty. Journal of Machine Learning Research, 18(26):1–5, 2017.

[19] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 5113–5120. IEEE, 2018.

[20] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1):23–33, 1997.

[21] Constantin Hubmann, Marvin Becker, Daniel Althoff, David Lenz, and Christoph Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 1671–1678. IEEE, 2017.

[22] Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. IEEE Transactions on Intelligent Vehicles, 3(1):5–17, 2018.

[23] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. Robotics science and systems: online proceedings, 2015, 2015.

[24] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation, 12(4):566–580, 1996.

[25] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Machine Learning, 49(2):193–208, Nov 2002.

[26] Cameron F. Kerry and Jack Karsten. Gauging investment in self-driving cars. Brookings, 2017.

[27] Minkyu Kim, Jaemin Lee, Steven Jens Jorgensen, and Luis Sentis. Social navigation planning based on people's awareness of robots. arXiv preprint arXiv:1809.08780, 2018.

[28] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P How. Real-time motion planning with applications to autonomous urban driving. IEEE Transactions on control systems technology, 17(5):1105–1118, 2009.

[29] Yiyuan Lee, Panpan Cai, and David Hsu. Magic: Learning macro-actions for online pomdp planning.

[30] Jing Liang, Utsav Patel, Adarsh Jagan Sathyamoorthy, and Dinesh Manocha. Realtime collision avoidance for mobile robots in dense crowds using implicit multi-sensor fusion and deep reinforcement learning. arXiv preprint arXiv:2004.03089, 2020.

[31] Michael H. Lim, Claire J. Tomlin, and Zachary N. Sunberg. Voronoi progressive widening: Efficient online solvers for continuous space MDPs and POMDPs with provably optimal components. In IEEE Conference on Decision and Control (CDC), 2021.

[32] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 6252–6259. IEEE, 2018.

[33] Yuanfu Luo, Panpan Cai, Aniket Bera, David Hsu, Wee Sun Lee, and Dinesh Manocha. Porca: Modeling and planning for autonomous driving among many pedestrians. IEEE Robotics and Automation Letters, 3(4):3418–3425, 2018.

[34] Cade Metz. The costly pursuit of self-driving cars continues on. and on. and on. The New York Times, 2021.

[35] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14424–14432, 2020.

[36] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. Journal of computational physics, 79(1):12–49, 1988.

[37] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. Mathematics of Operations Research, 12(3):441–450, 1987.

[38] Stefania Pellegrinelli, Henny Admoni, Shervin Javdani, and Siddhartha Srinivasa. Human-robot shared workspace collaboration via hindsight optimization. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 831–838. IEEE, 2016.

[39] Adarsh Jagan Sathyamoorthy, Jing Liang, Utsav Patel, Tianrui Guan, Rohan Chandra, and Dinesh Manocha. Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 11345–11352. IEEE, 2020.

[40] Bill Schiller, Vassilios Morellas, and Max Donath. Collision avoidance for highway vehicles using the virtual bumper controller. In Proceedings of the IEEE International Symposium on Intelligent Vehicles, pages 28–30. IEEE, New York, 1998.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[42] James Albert Sethian. Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, volume 3. Cambridge university press, 1999.

[43] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In Advances in Neural Information Processing Systems, pages 2164–2172, 2010.

[44] Weilong Song, Guangming Xiong, and Huiyan Chen. Intention-aware autonomous driving decision-making in an uncontrolled intersection. Mathematical Problems in Engineering, 2016, 2016.

[45] Ke Sun, Brent Schlotfeldt, George J Pappas, and Vijay Kumar. Stochastic motion planning under partial observability for mobile robots with continuous range measurements. IEEE Transactions on Robotics, 37(3):979–995, 2020.

[46] Zachary Sunberg and Mykel J. Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In International Conference on Automated Planning and Scheduling, 2018.

[47] Zachary N. Sunberg, Christopher J. Ho, and J. Kochenderfer, Mykel. The value of inferring the internal state of traffic participants for autonomous freeway driving. In American Control Conference (ACC), 2017.

[48] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. Journal of field Robotics, 23(9):661–692, 2006.

[49] Alberto Valero-Gomez, Javier Gómez, Santiago Garrido, and Luis Moreno. Fast marching methods in path planning. IEEE Robotics & Automation Magazine, 20:111 – 120, 12 2013.

[50] Jur Van Den Berg, Pieter Abbeel, and Ken Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. The International Journal of Robotics Research, 30(7):895–913, 2011.

[51] Dizan Vasquez, Thierry Fraichard, and Christian Laugier. Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. The International Journal of Robotics Research, 28(11-12):1486–1506, 2009.

[52] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. Journal of Artificial Intelligence Research, 58:231–266, 2017.