ASEN 6519 - Decision Making Under Uncertainty
Final Project Report
Intention Aware Online POMDP Planning for Autonomous Driving in Dynamic Environment with Simulated Crowd

Himanshu Gupta
Department of Computer Science
University of Colorado Boulder
Email: Himanshu.Gupta@colorado.edu

Dhanendra Soni
Department of Computer Science
University of Colorado Boulder
Email: Dhanendra.Soni@colorado.edu

Zachary Sunberg
Department of Aerospace Engineering
University of Colorado Boulder
Email: zachary.sunberg@colorado.edu

***Abstract*—This project report presents our attempt at replicating the famous research paper on intention-aware online planning approach for autonomous driving amid many pedestrians. To drive near pedestrians safely, efficiently, and smoothly, autonomous vehicles must estimate unknown pedestrian intentions and hedge against the uncertainty in intention estimates in order to choose actions that are effective and robust. A key feature of this approach is to use the partially observable Markov decision process (POMDP) for systematic, robust decision making under uncertainty. POMDPs are PSPACE-complete problems and so there are concerns about the potentially high computational complexity of POMDP planning. However, experiments have shown that online POMDP based planners generate optimal policies in near real time in a complex, dynamic environment. This indicates that POMDP planning is improving fast in computational efficiency and is becoming increasingly practical as a tool for robot planning under uncertainty.**

1. Introduction

A perfect autonomous system that can navigate in a cluttered dynamic environments is extremely hard to achieve due to uncertainties in the dynamic environment and inaccuracies of sensor measurements. A reliable robotic system in such dynamic environment needs to handle high level of uncertainties in the environment and then generate an optimal plan and control mechanism.

The Partially Observable Markov Decision Process (POMDP) is a mathematical tool that can efficiently hedge against uncertainties to provide reliable, robust and optimal decisions or plans or control. In this work, we are reproducing the work of Bai et. al.[1] for successful navigation of an autonomous cart in a dynamic environment using online POMDP planners.

There are many successful autonomous vehicles today, but it is still uncommon to see autonomous vehicles driving among many pedestrians. Social Navigation is still an open problem and a lot of research is being done in this field.

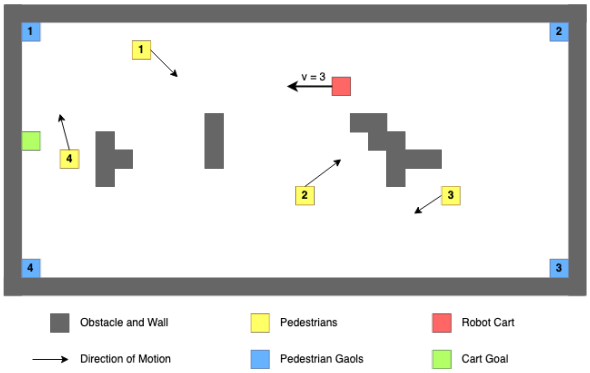


Figure 1. An autonomous golf cart driving in a crowded, dynamic environment.

To drive near pedestrians safely, efficiently, and smoothly, autonomous vehicles must estimate unknown pedestrian intentions and hedge against the uncertainty in intention estimates in order to choose actions that are effective and robust. They must also reason about the long-term effects of immediate actions. Simple reactive control methods are inadequate here. They just choose actions based on sensor data on the current system that seem best for the current state of the system, and is thus greedy. This often results in overly aggressive or conservative actions, or attractive short-term actions with undesirable longer-term consequences.

POMDP planning provides a systematic approach to overcome these issues. However, the use of POMDPs for robot planning under uncertainty is not widespread. POMDPs are PSPACE-complete and so getting a complete solution to POMDP problems in real time is computationally not feasible. Offline methods for solving POMDPs have been used before for intention-aware motion planning[2]. The earlier work builds a discrete-state POMDP model and solves the model offline for a policy. With recent advancements in online POMDP planning techniques, POMDPs with continuous state space and large discrete observation space can now be solved real time with algorithms like

DESPOT and POMCP.

We constructed a POMDP model for autonomous driving in a dynamic environment with pedestrians. The POMDP model captures the uncertainty in pedestrian intentions and maintains a belief over pedestrian goals. In this work, we applied DESPOT [3], a state-of-the-art approximate online POMDP planning algorithm, to autonomous driving in a complex, dynamic environment (Fig. 1). We have also applied POMCP algorithm to our POMDP and compared its results with DESPOT. By exploiting online POMDP planning, this work made several advances. First, online planning allows us to handle dynamic environment changes, e.g., sudden appearance of new obstacles, while the earlier work assumes a fixed environment. Second, both DESPOT and POMCP algorithms allow continuous state variables in POMDP models, which enables us to model vehicle and pedestrian dynamics more accurately and conveniently. Third, due to the limitation on computational efficiency, the earlier algorithm computes the action for each pedestrian in isolation and then chooses the most conservative one. These algorithms scale much better and allows for a more realistic model that treats the pedestrians jointly and captures their interactions.

We run everything in simulations on a discrete grid world with hand designed complex human motion trajectories. A brief overview of the approach is presented in Section 3 of the report. Experimental setup and results are presented in Section 5. Refer to our project github page [4] for simulation gifs of the autonomous cart moving in a dynamic environment.

2. Related Work

A POMDP can either be solved offline or by doing online search. In offline planning, we compute beforehand a policy for all possible future scenarios, and the robot executes the computed policy based on the sensor data received[14]. This approach is not suitable for very large discrete state or continuous state POMDPs due to exponential number of possible scenarios. It gets even more difficult when there are dynamic elements in the environment, as it is extremely difficult to model dynamic elements by predicting their expected behaviour in future. On the other hand, online planning interleaves planning and plan execution [6]. We maintain a belief distribution over all the possible states. The robot searches for the best action for just the current belief, executes that action, and updates its belief distribution. The process is then repeated at the new belief just generated. Online algorithms apply several techniques for approximations and computational efficiency, including heuristic search, branch-and-bound pruning, and Monte Carlo sampling [7]. AEMS[10], POMCP [8] and DESPOT [9] were the first few online POMDP algorithms. Both POMCP and DESPOT can handle large number of states, but DESPOT has a much stronger worst-case performance bound. POMCPOW [11] and DESPOT-α [12] are amongst the best online POMDP algorithms available today.

Human life to autonomous vehicles

Figure 2. An apt meme that displays the seriousness and need for safety in autonomous vehicles.

Nowadays, there are many successful systems for autonomous driving [13],[14]. It is a really exciting field of robotics. However, these systems have paid relatively little attention to close interaction with pedestrians, especially, in unstructured environments. Such technology is useful for providing on demand, personalized transportation in densely populated urban localities, such as university campuses, business parks. These systems can act as assistive nurses in hospitals and carry medical tools and medicines to different patients in the hospital. Since we expect these systems to navigate in close human proximity, they should always ensure safe motion planning (Fig. 2).

One main difficulty of autonomous driving among pedestrians is to incorporate pedestrian intentions and behaviors into decision making. There are two related, but orthogonal issues here. One is pedestrian intention and behavior modeling. There are various modeling approaches based on, e.g., linear dynamic systems with Gaussian noise, hidden Markov models (HMMs) [15], [16], Gaussian processes (GPs) [17]. Recent work has also been done using neural networks for generating such human models [18].

However, this work focuses on the orthogonal issue, decision making, which determines the vehicle action given a predictive pedestrian behavior model. The simplest approach is reactive control [19], [20]. It does not require a sophisticated predictive model and basically ignores prediction uncertainty during decision making. It is fast, but since it is just maximising its immediate reward, it often results in suboptimal decisions over the long term. A more sophisticated approach is to compute an open-loop plan, i.e., a path, using a predictive model and then execute the plan with a feedback controller [16], [21], [22]. Recent work on social navigation using Deep Reinforcement Learning [23] has shown lots of potential.

POMDP planning goes one step further: it reasons about uncertainties systematically and computes a close-loop plan that handles all future contingencies. In addition to human behavior uncertainty, the POMDP approach can also incorporate vehicle control and sensing uncertainties into decision

making systematically. Conceptually, POMDP planning is analogous to computing an optimal controller for a linear quadratic Gaussian (LQG) system, but POMDP planning is more general and does not assume linear dynamics or Gaussian noise distribution. The advantages of POMDP planning comes at the cost of higher computational complexity. This work is an attempt to showcase that POMDP planning is improving fast in computational efficiency and is becoming more practical as a tool for robot planning under uncertainty. Of course, it does not replace other approaches. The trade-off between the optimality of decision making and computational efficiency remains, but the gap is narrowing.

3. Overview

For any autonomous vehicle that drives in an environment with pedestrians in it, it needs to infer the pedestrian's intended goals and also take into consideration the uncertainty in the estimation of these intended goals. We usually represent this uncertainty in terms of belief, a probability distribution over all possible intentions. We also need to consider the long term effects of current action for robust and safe planning and choose actions that maximise some chosen expected reward.

The original paper gives an amazing example. Suppose the vehicle is approaching a pedestrian walking on the sidewalk(Fig. 3). For simplicity let's consider there are two actions for pedestrian, it can either keep walking straight and stay on the sidewalk or can turn left and decide to cross the road. If the pedestrian stays on the sidewalk, vehicle can accelerate and pass quickly and if the pedestrian decided to cross the road, vehicle need to slow down or break. Let's denote the belief over these two pedestrian intentions by $(p, 1-p)$ where p denotes the probability of pedestrian staying on the sidewalk. Let's also assume that the initial belief is $(0.51, 0.49)$. If we take the decision based on maximum likelihood then the vehicle can accelerate and pass. But at the next moment, sensors receives a new input indicating that the pedestrian is getting off the sidewalk and trying to cross the road. With this new observation the belief is updated to $(0.3, 0.7)$ and now vehicle breaks to slow down or stop. Since the vehicle has accelerated in the previous step and driving at a higher speed now, it would be difficult to break immediately and it might result in an accident. The maximum likelihood decision making is greedy in nature and does not consider the long-term effects of the current action. The self-driving Uber car that accidentally killed a woman wasn't able to recognize that pedestrians jaywalk. As a result, it couldn't correctly predict her path and concluded that it needed to brake just 1.3 seconds before it struck her. The vehicle didn't take into consideration uncertainty in its human intention estimate or the effects of future sensor observations on the belief while choosing its actions.

This limitation is common in greedy decision-making approaches like reactive control. On the other hand, POMDP planning is best suitable for scenarios when we need to consider long-terms effects in a structured way. The POMDP planner searches a belief tree rooted at the current belief

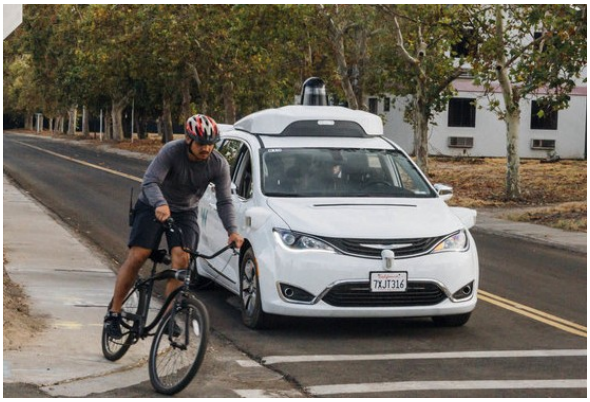


Figure 3. Waymo's Chrysler Pacifica minivan trying to figure out if the cyclist will go straight or turn left. Julia Wang/Waymo via AP.

(Fig. 5) and branches on all actions and observations. The POMDP planning is costly, as in the worst case, the tree has an exponential size $\mathcal{O}(|A|^H|Z|^H)$, where $|A|$ is the action space size $|Z|$ is the observation space size and H is the maximum tree height. The action space and the observation space are generally huge for autonomous vehicles.

The planning task is divided into two levels for computational efficiency. The high level planner controls the vehicle path and steering. It tries to find the minimum cost path from source to target while avoiding collisions and considering predicted human motions as well. This level of planning is handled by A^* path planner. The low level planner controls the vehicle speed along the planned path. We use POMDP to model the uncertainties in pedestrian intentions. For online POMDP planning, we use DESPOT algorithm for two major reasons. Since it only needs a generative model, it is easy to code and it is capable of handling the large observation spaces efficiently.

In this work, we have divided the entire task into three components: a path planner, a POMDP speed planner and a belief tracker. The path planner plans an optimal path considering the current belief. The belief updater maintains a belief over system states and updates the belief at each time step based on observations and vehicle actions. The POMDP planner computes a conditional plan for the next H steps and uses the best estimated action at that belief state to modify the speed of the vehicle. Since the environment is dynamic, both the high level and the low level planners are re-planned at each time step.

4. Intention Aware Online POMDP Planning

This section covers all the technical details of this work and the simplifications we made to the original POMDP for our simulation environment. We first describe the basic structure of any POMDP problem and then give details for our POMDP. This is followed by specifying the A^* path planner, brief overview of DESPOT and module for belief update.

4.1. POMDP Basics

A POMDP is a system that takes a sequence of actions under uncertainty in order to maximize its total expected reward. The formal definition of a POMDP model is given as 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.

At each time step, the system takes an action $a \in A$ in state $s \in S$ to reach state $s' \in S$ and then get an observation $z \in Z$. The transition model T is specified by a conditional probability function $T(s, a, s') = p(s'|s, a)$ which specifies the probability distribution of the new state s' when the system takes an action a in state s. Similarly the observation model O is specified by a conditional probability function $O(s', a, z) = p(z|s', a)$ which specifies the probability distribution of observation z. In real world this models the noisy sensor observations. The Reward model R is specified by a function $R(s, a)$ which specifies the immediate reward of taking action a in state s.

The system state is not completely known in a POMDP, and so the system maintains a belief over all the possible states. Let b_{t-1} be the belief at time $t-1$ and if the system takes action a_t and gets an observation z_t at the next time t, then we apply the Bayes' rule to calculate the new belief b_t as:

$$b_t(s') = \eta O(s', a_t, z_t) \sum_{s \in S} T(s, a_t, s') b_{t-1}(s) \qquad (1)$$

where η is a normalization constant. This defines a belief update function τ such that $b_t = \tau(b_0, a_1, z_1, ..., a_t, z_t)$, where b_o is the initial belief.

A POMDP system returns a policy which is a function π that specifies the action $a = \pi(b)$ at belief b. In online POMDP system we want a policy that maximizes the value which is the expected total reward at the current belief b:

$$V_\pi(b) = \mathrm{E}\big(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t))|b_o = b\big) \qquad (2)$$

where $\gamma \in (0, 1)$ is a discount factor, which specifies the preference of immediate goals over future goals.

4.2. Our simplified POMDP Model

This subsection describes the structure of state, observation and reward of my POMDP.

4.2.1. Vehicle Modeling. The vehicle(cart) state consists of position (x, y), orientation θ, current speed v and the generated A^* path. In our simulation environment, the orientation of the cart has just four possible values: 0, 90, 180 and 270. θ is determined using the cart's current position and previous position. We keep track of this using the generated $A*$ path. The action space of our POMDP model has three discretized actions $[-1, 0, 1]$ representing $Decelerate$, $Maintain$ and $Accelerate$ respectively. Given an action, we update cart's velocity and move it along the generated path by that many steps considering Δt as 1.

4.2.2. Pedestrian Modeling. The pedestrian state consist of position (x_i, y_i) and goal g_i. Each pedestrian is trying to reach a goal in the environment and typically a goal is a position (x_g, y_g) in the environment. This goal is the intention that the vehicle doesn't know before hand and tries to infer it using observations. We assume that each pedestrian walks one step towards it's goal in one time step in it's desired path. There are four pedestrians and four goals in our environment and for simplicity every human is traveling to a different goal.

4.2.3. Observations. An observation in our model is a vector consisting of positions of all pedestrians. We do not model observation noises for these variables, as they are generally small because of significant improvements in sensors and do not affect decision making substantially. However, there are no direct observations on pedestrian intentions, and they are the key partially observable variables in our model. We must infer the pedestrian intentions from the observations received over time, and also hedge against the estimation uncertainty during decision making. In our POMDP generative model, we have considered uncertainty in our human intention estimate by assuming that the human will move towards its inferred goal 70% of the time and will move to any of the other three goals randomly 10% of the time.

4.2.4. Reward Modeling. The reward model guides the cart towards an optimal driving behaviour which is safe, collision-free, efficient and reaches the goal. We have three different types of rewards that we considered in our model. They are:

- Collision Reward: If the cart gets within a certain threshold distance to a pedestrian or an obstacle, then there is a large penalty of R_{col}. This reward is modelled to ensure safety on pedestrians as well as the cart.
- Goal Reward: If the cart gets close to the goal within a certain threshold distance than there is a large reward of R_{goal}. This reward is modelled to encourage the cart to reach it's goal.
- Speed Reward: If the Cart is driving slower than it's max possible speed then there is a small proportional penalty $R_{speed} = (v - v_{max})/v_{max}$. This reward is modelled to encourage the cart to drive fast whenever possible.

4.3. A* Path Planner

A^* is a path planning algorithm that finds a path from a given starting node to a desired goal node while minimizing a specified cost function. The general formulation of cost function is,

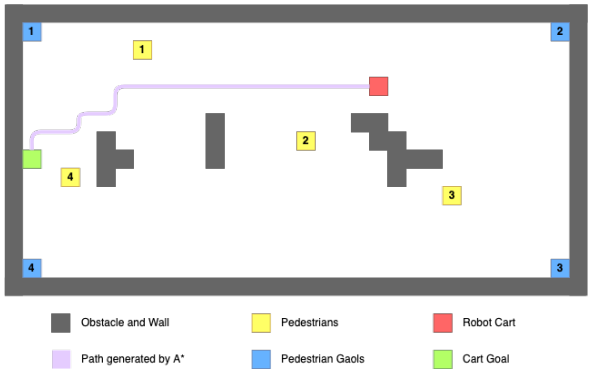


Figure 4. Path generated by A^* algorithm at time step t

$$f(n) = g(n) + h(n)$$

where n is the next node in the path, $g(n)$ is the cost of the path from start node to the next node n and $h(n)$ is a heuristic function that estimates the cost of optimal path from n to the goal node.

For simplicity purposes, at any time step t, we have modeled the pedestrian as an obstacle at its position. We designed a smart heuristic function to guide A^* to find a path for the cart to its goal state quickly.

The path from cart to goal is represented by a sequence of grid points (x_i, y_i) where each grid point in the path is adjacent to it's previous and successive points. We define the cost C of the path as sum of the following components,

$$C(p) = \sum_{i=0}^{n} C_{obs}(x_i, y_i) + \sum_{i=0}^{n} C_{safe}(x_i, y_i) + \sum_{i=1}^{n} C_{move}(i)$$

The first component adds the cost for collision with obstacles, second component adds the cost for safety around the obstacles and third component is pushing for the shortest path.

- $C_{obs}(x_i, y_i)$ is the cost for each obstacle cell in the environment grid world. Every obstacle has a high cost of 99 associated with it.
- $C_{safe}(x_i, y_i)$ is the cost for each grid cell that is adjacent to the obstacles. It can be visualized as padding and every padded cell has a cost of 50. This prevents the cart from going too close to the obstacles if there are better alternatives. It is important that the cart maintains a safe distance from the obstacles and pedestrians in the environment.
- $C_{move}(i)$ is the cost associated with every step of the cart in the environment. Its a considerably low cost of value 1 and it guides the cart to find the shortest path possible.

The generated optimal path is passed to the speed planner module that determines the best possible speed to use while moving forward through the generate path at that time step.

4.4. DESPOT

To choose the best vehicle action at the current belief b_0, we perform online POMDP planning. We search a belief tree with root at b_0 and maximum height H. Each tree node represents a belief, and each edge represents an action-observation pair. If a child node b' is connected to its parent b by an edge (a, z), then $b' = \tau(b, a, z)$ according to (1). To search a belief tree, we perform a post-order traversal. At each leaf node, we simulate a default policy to obtain a lower bound on its value. At each internal node, we apply Bellman's principle of optimality to choose the best action:

$$V(b) = \max_{a \in A} \Big\{ \sum_{s \in S} b(s) R(s, a) + \gamma \sum_{z \in Z} p(z/b, a) V(\tau(b, a, z)) \Big\} \tag{3}$$

which recursively computes the maximum value of action branches and the average value of observation branches. The result is an approximately optimal policy π for the current belief b_0. The vehicle then executes the first action of the policy, $\pi(b_0)$. The idea is straightforward but since the tree grows on the order of $O(|A|^H |Z|^H)$, it is impractical to construct or search the full tree when the action space or the observation space is large.

To overcome this challenge, we use DESPOT, whose key strengths include handling large observation spaces. Intuitively, we do not need to examine all observation branches to identify an approximately optimal action, because the second sum in (3) computes an average value over observation branches. 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. It corresponds to a particular sequence of action chosen by the policy and observation received. We now define a subtree, called DEterminized Sparse Partially Observable Tree (DESPOT), which contains only the belief-tree nodes and edges traversed by all policy under the sampled scenarios. A DESPOT tree is a sparsely sampled belief tree. While the original belief tree contains $O(|A|^H |Z|^H)$ nodes, the DESPOT tree contains only $O(|A|^H K)$ nodes, leading to dramatic improvement in computational efficiency for moderate K values.

DESPOT is an anytime algorithm, which builds its tree incrementally by performing heuristic search guided by a lower bound and an upper bound on the value at each tree node. For the lower bound at a leaf node b_l , we use the empirical value of the default policy of always accelerating under the sampled scenarios. The value of this policy under a set of sampled scenario can be easily calculated by simulation. For the upper bound at b_l, it is computed as the average of the upper bounds for the scenarios. There are two cases for computing the upper for a scenario. If the vehicle and a pedestrian collides under the scenario, the bound is R_{col}. Otherwise, it is $\gamma^t R_{goal}$, where t is the minimum number of steps needed by the cart to reach the goal, assuming that the vehicle drives a maximum speed with no pedestrians

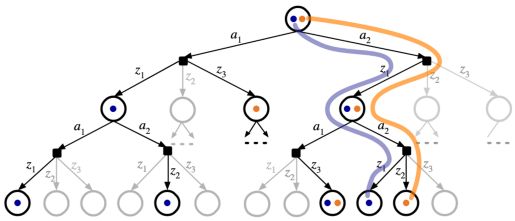


Figure 5. A belief tree of height H= 2(gray) and a corresponding DESPOT tree (black) obtained with 2 sampled scenarios, shown in blue and orange.The blue and orange curves indicate the execution paths of a same policy under the two scenarios.

around. We then take the average over the sampled scenarios to obtain the upper bound at b_l. For the internal nodes of the DESPOT tree, we evaluate their lower and upper bounds recursively using (3). The speed planner computes a policy over the entire DESPOT tree to hedge against uncertainty, but we only care about the first step of the policy, the action at the root of tree.

4.5. Belief Tracker

The partially observable variables in our system are pedestrian intentions which are inferred by the belief tracker based on the series of observations received. Since the number of pedestrians and their goals are fixed in our system, the number of intentions are finite and belief over all the intentions form a discrete probability distribution. For each pedestrian, the belief tracker observes its movement from (x, y) to (x', y') and updates the belief $b(g)$ over all the possible intentions/goal locations to $b'(g)$using the following update formula: $b'(g) = \eta p(x', y'|x, y, v, g)b(g)$, where η is a normalization constant. For simplicity, we have assumed $p(x', y'|x, y, v, g)$ to be just the reciprocal of the euclidean distance between (x', y') and goal g.

The change in pedestrian's intention is captured by the belief tracker. It tracks the belief over all possible goals for each pedestrian in the environment. When a pedestrian changes it's goal from g_1 to g_2, then for that pedestrian, the belief $b(g_1)$ decreases and the belief $b(g_2)$ increases.

5. Experimental Details and Results

5.1. Experimental Details

We performed all our experiments in simulation. We designed our simulation environment as a grid world in python and used Julia for its inbuilt POMDP solvers.

The Custom Grid World Environment is implemented in Python and it has three major components: the Grid World implementation, pedestrians model and the A^* algorithm.

- The Grid World Environment: We create a simple Grid World environment of size 16x32 as shown in Fig. 6. The outer boundary of the environment is modeled as the obstacle (outer wall) and there are 3 other obstacles of different shapes in the environment. The Cart is starting at the right wall from coordinate $(8, 30)$ and trying to reach the goal at the left wall at $(7, 1)$.
- Pedestrian Model: We have initialized 4 pedestrian in the environment and they are moving to reach their respective goals. The goals for the pedestrians are placed at the corners of the environment. For simplicity, the actual goal of the human is always assumed to be the same. If we number the pedestrians and goals from top left corner in clockwise fashion starting at number one, then pedestrian-1 is going to goal-3, pedestrian-2 is going to goal-4, pedestrian-3 is going to goal-2 and pedestrian-4 is going to goal-1. Pedestrians are modeled to avoid the obstacles in their paths while moving to their goals.
- A^* Algorithm: We have taken inspiration from [24] to implement the A^* algorithm. The core logic of the A^* is implemented in C++ in this repository. We have used the same implementation and modified the cost distribution in the environment to fit our use case. The optimal path it returns is a list of grid positions on the path.

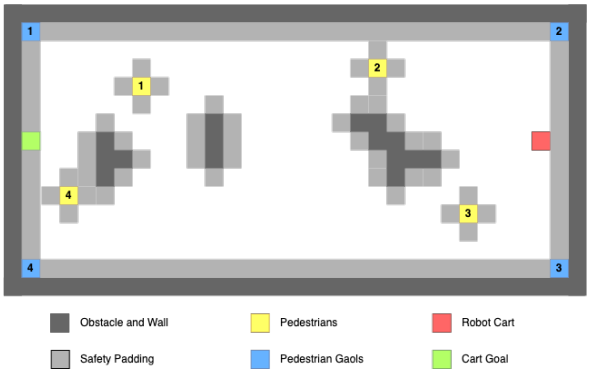


Figure 6. Grid Environment

The POMDP model is implemented in Julia. We created our own POMDP model as mentioned in Section 4. We designed a generative model that takes in the current state and an action and returns the next state, an observation and a reward. We used standard DESPOT solver with custom implementation of Lower Bound and Upper Bound functions to solve our POMDP. We also used the standard POMCP solver with custom parameters and compared their performance.

We created 3 pedestrian motion trajectories with different level of complexities. The complexity of these trajectories is described by the potential overlap of path between pedestrians and the cart. The trajectories are numbered in increasing level of complexity from 1 to 3 and are used in experiments 1 to 3 respectively.

We performed several runs of DESPOT and POMCP algorithms over these trajectories and reported the average time taken to reach the goal and average reward in each of these experiment.

5.2. Results

The results of the three experiments are presented in Table 1. DESPOT outperformed POMCP in the first experiment which has the least complex trajectory of pedestrian motion. DESPOT also outperformed POMCP in the third experiment which has the most complex pedestrian motion trajectory. DESPOT has a much stronger worst-case performance bound and same can be observed from the experiment results. On the contrary, POMCP performed better than DESPOT in the second experiment which has a moderately complex pedestrian trajectory.

	DESPOT		POMCP	
	Time to Goal	Reward	Time to Goal	Reward
Experiment #1	11	195.645	7	-152.914
Experiment #2	11	101.829	10	260.283
Experiment #3	19	-21.488	13	-30.710

TABLE 1. REWARDS FOR THREE DIFFERENT EXPERIMENTS USING DESPOT AND POMCP

6. Future Work

In this work we are using a POMDP planner to control the speed of the cart and a separate planner is handling the navigation and path planning task. In future, we plan to control the steering angle of the the cart using a POMDP planner to make a more robust driving cart. Currently we have implemented the custom grid world in Python. We plan to extend that implementation to Julia on a continuous state space for better integration, faster performance and to deal with more realistic dynamics.

7. Conclusion

This work presents an intention aware online POMDP planner for autonomous driving in complex dynamic environment. Our experiments show that POMDP is a powerful tool for autonomous driving in complex environments with uncertainties. This work provides two important lessons. First, a principled decision framework such as POMDP can be used for safe, efficient, and smooth autonomous driving. Second, despite the concerns about the complexity of POMDP planning, it is improving fast in computational efficiency and becoming increasingly practical as a tool for robot planning under uncertainty. We believe that with rapid progress in development and improvement of online POMDP planners, formulating and solving autonomous driving as a POMDP will become inevitable.

Acknowledgments

The students would like to thank professor Zachary Sunberg for his constant guidance and tech support for Julia. It wouldn't have been possible without his help.

References

[1] Bai, Haoyu, et al. "Intention-aware online POMDP planning for autonomous driving in a crowd." 2015 ieee international conference on robotics and automation (ICRA). IEEE, 2015.

[2] T. Bandyopadhyay, K. Won, E. Frazzoli, D. Hsu, W. Lee, and D. Rus, "Intention-aware motion planning," in Algorithmic Foundations of Robotics X—Proc. Int. Workshop on the Algorithmic Foundations of Robotics (WAFR), 2012.

[3] A. Somani, N. Ye, D. Hsu, and W. Lee, "DESPOT: Online POMDP planning with regularization," in Advances in Neural Information Processing Systems (NIPS), 2013.

[4] https://github.com/himanshugupta1009/autonomous_golf_cart/

[5] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," Autonomous Agents and Multi-Agent Systems, vol. 27, no. 1, pp. 1–51, 2013.

[6] R. He, E. Brunskill, and N. Roy, "Efficient planning under uncertainty with macro-actions," J. Artificial Intelligence Research, vol. 40, no. 1, pp. 523–570, 2011.

[7] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs," J. Artificial Intelligence Research, vol. 32, no. 1, pp. 663–704, 2008

[8] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in Advances in Neural Information Processing Systems (NIPS), 2010.

[9] A. Somani, N. Ye, D. Hsu, and W. Lee, "DESPOT: Online POMDP planning with regularization," in Advances in Neural Information Processing Systems (NIPS), 2013.

[10] Ross, Stephane, Joelle Pineau, and Brahim Chaib-draa. "Theoretical analysis of heuristic search methods for online POMDPs." Advances in neural information processing systems. 2008.

[11] Sunberg, Zachary N., and Mykel J. Kochenderfer. "Online algorithms for POMDPs with continuous state, action, and observation spaces." Twenty-Eighth International Conference on Automated Planning and Scheduling. 2018.

[12] Garg, Neha P., David Hsu, and Wee Sun Lee. "DESPOT-α: Online POMDP Planning With Large State And Observation Spaces." Robotics: Science and Systems. 2019.

[13] M. Montemerlo et al., "Junior: The Stanford entry in the urban challenge," J. Field Robotics, vol. 25, no. 9, pp. 569–597, 2008.

[14] C. Urmson et al., "Autonomous driving in urban environments: Boss and the urban challenge," J. Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[15] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," Int. J. Robotics Research, vol. 24, no. 1, pp. 31–48, 2005

[16] D. Vasquez, T. Fraichard, and C. Laugier, "Growing hidden Markov models: An incremental tool for learning and predicting human and vehicle motion," Int. J. Robotics Research, vol. 28, no. 11–12, pp. 1486–1506, 2009.

[17] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots Systems, 2008

[18] Kratzer, Philipp, Marc Toussaint, and Jim Mainprice. "Prediction of Human Full-Body Movements with Motion Optimization and Recurrent Neural Networks." arXiv preprint arXiv:1910.01843 (2019).

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

[20] B. Schiller, V. Morellas, and M. Donath, "Collision avoidance for highway vehicles using the virtual bumper controller," in Proc. IEEE Int. Symp. on Intelligent Vehicles, 1998.

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

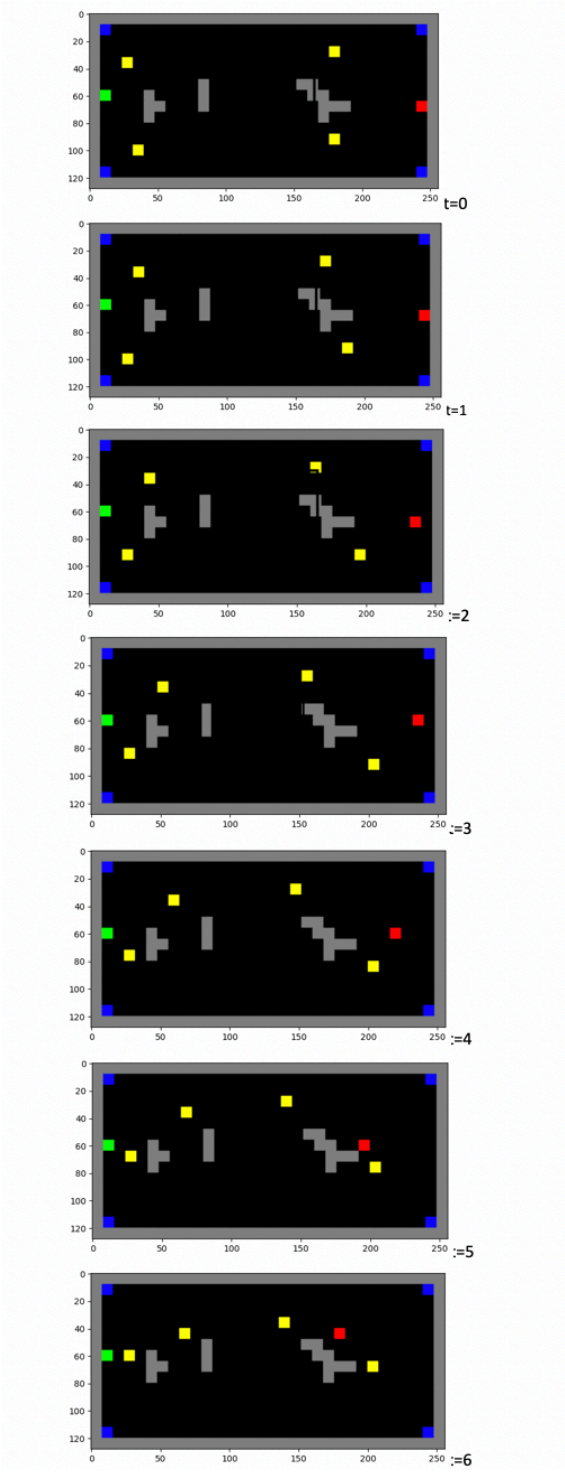
[22] J. van den Berg, P. Abbeel, and K. Goldberg, “LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information,” Int. J. Robotics Research, vol. 30, no. 7, pp. 895–913, 2011

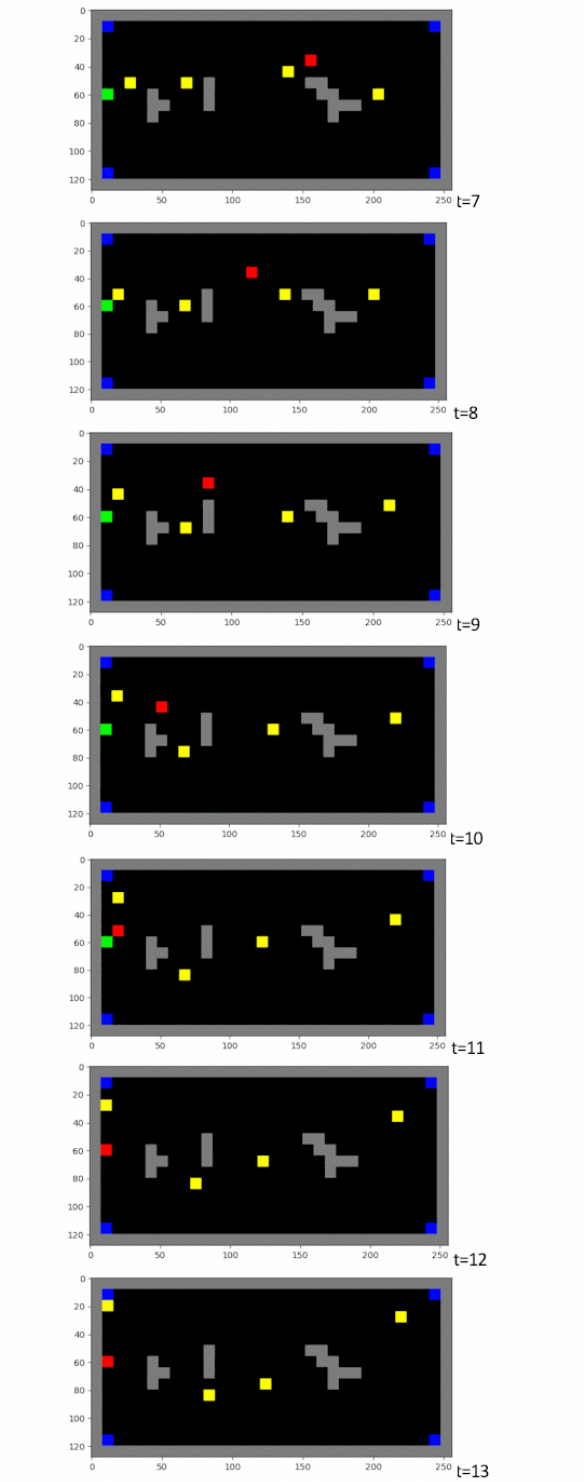
[23] Chen, Yu Fan, et al. ”Socially aware motion planning with deep reinforcement learning.” 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017.

[24] https://github.com/hjweide/a-star

Appendix

Environment snapshots at each time step of DESPOT Algorithm execution





Cart Cart Goal Pedestrians Pedestrian Goals Obstacles