Dynamic Obstacle Avoidance in Shared Human Robot Workspace

Kaleb Bishop
Department of Computer Science
University of Colorado Boulder
Boulder, USA

Himanshu Gupta
Department of Computer Science
University of Colorado Boulder
Boulder, USA

Dylan Kriegman
Department of Computer Science
University of Colorado Boulder
Boulder, USA

Tuhina Tripathi
Department of Computer Science
University of Colorado Boulder
Boulder, USA

Yi-Shiuan Tung
Department of Computer Science
University of Colorado Boulder
Boulder, USA

Nikolaus Correll
Department of Computer Science
University of Colorado Boulder
Boulder, USA

***Abstract*—Collaboration between humans often leads to a decrease in the total completion time of a task. This can be extended to collaboration between a human and a robot on a common task in a shared workspace. During human-robot collaboration, a robot and a user must often complete a disjoint set of tasks that use an overlapping set of objects, without using the same object simultaneously. A key challenge is deciding what task the robot should perform next in order to facilitate fluent and efficient collaboration. In this paper, we briefly describe a prior system for inferring a probability distribution over human goals, and provide an approach to produce action for the manipulator given that distribution in real time. We propose to achieve this by formulating the problem as a Partially Observable Markov Decision Processes (POMDPs) and solving this POMDP in an online fashion using state of the art tree search techniques that can be guided by a Probabilistic Roadmap. To overcome the integration issues mentioned in the paer, we implemented a PRM-based approach in the simulation environment and showed that it's effective in avoiding collisions while ensuring that the robot accomplishes its task.**

I. Introduction

Collaboration between a human and a robot (HRC) has been widely used in many real-world scenarios ranging from autonomous driving to industrial manufacturing. Automation has revolutionized the manufacturing process over the last few decades. There are so many manufacturing tasks that are tedious or strenuous for humans to perform. Some of these tasks, such as automobile assembly, are difficult to completely automate because they require workers to collaborate in close proximity and adapt to each other's decisions and motions. Robots cannot currently do such complex tasks. Rather than completely automating such tasks, designing robotic systems that can interact in the same environment as humans and assist them in finishing a task seems more feasible. We believe safe human robot collaboration can enable effective task execution while reducing both strain of the human and total time for task completion.

HRC requires the human and robot to execute tasks in a shared workspace, in order to study the interactions between them. Consider having a set of actions to be performed e.g., to open, to grasp, to move (tasks) and objects to perform the action on e.g., a door, a glass, a cup (goals), the human can select tasks to perform and goals on which to perform them. Meanwhile, the robot can assist the human by performing a different task on the same goal (e.g. grasp a bottle if the human wants to open it) or on another goal (e.g., to open the fridge door if the human wants to grab a bottle from it). The robot can also support the human by picking an uncorrelated task on a secondary goal. One of the goals of HRC is to guarantee human satisfaction and comfort in the approach taken, therefore, the human has priority over the robot in selecting the task. Thus, the robot must adapt to the human's task selections. Robot adaptation is even more complex with certain constraints on the execution order of the tasks (e.g. in order to grasp the bottle in the fridge, the fridge door must be open) and when the robot is sharing workspace, goals and/or tasks with more than one human.

Fig. 1: The human and manipulator working together to finish the task of clearing the table top.

In order for a robotic system to behave safely in a dynamic environment with humans, it must try to estimate the unknown human intentions and plan its motion accordingly. A popular approach to maintain safety in the presence of moving humans is to interleave planning and execution within a motion planning framework. Humans are treated as dynamic obstacles and the human motion is predicted in the replanning window

by using a bounded velocity model. A trajectory is generated using a cost function and that trajectory is implemented for a small duration. This set of steps is repeated after every small time interval.

We believe that there are three major drawbacks to these methods. Firstly, they assume that their predicted human motion is correct and don't hedge against the uncertainty in human motion prediction. Secondly, they don't take into consideration the long-term effects of immediate robot arm's actions, and is thus greedy. This often results in overly aggressive or conservative actions, or attractive short-term actions with undesirable longer-term consequences. Thirdly, they don't take into consideration the effects of robot arm's motion on human's motion, which is definitely not the case in real life. Modeling this can give rise to two kinds of behaviour. In the first behaviour, the robot arm can try to affect the human's motion as less as possible. In the second behaviour, the robot arm can attempt to modify the human motion and guide it to a more optimal action (if there exists one) in a collaborative task.

To overcome these drawbacks, we propose modeling the problem as a Partially Observable Markov Decision Process or POMDP [1]. A POMDP is a mathematical tool that can efficiently hedge against uncertainties to provide reliable, robust and optimal decisions. 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 [3] and DESPOT-α [4].

In this project, we have proposed to perform a human robot collaborative task of clearing a table top and solve it by formulating it as a POMDP. Our focus right now is mostly on decision making under uncertainty and not on human modeling. We believe our two major contributions in this work, when finished, are to formulate the tabletop clearing experiment as a POMDP and creating a system that can do complex collaborative tasks.

II. RELATED WORK

A POMDP can be solved using both offline and online sampling-based approaches. In offline planning, we compute a policy for all possible future scenarios beforehand, and the robot executes the computed policy based on the sensor data received [5]. 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 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 [9], POMCP [8] and DESPOT [3] 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 [10] and DESPOT-α [4] are amongst the best online POMDP algorithms available today.

One main difficulty of autonomous behavior in human robot collaboration tasks is to incorporate human intentions and behaviors into decision making. There are two related, but orthogonal issues here. One is human intention and behavior modeling. There are various modeling based approaches, e.g., linear dynamic systems with Gaussian noise, hidden Markov models (HMMs) [11], [12], Gaussian processes (GPs) [13]. Recent work has also been done using neural networks for generating such human models [14]. Authors in [15] proposed a model based on radial basis function neural networks (RBFNN) to predict human intentions, which requires the interaction force of human limb and robot arm. [16] uses a set of probabilistic state machines, each regarding to a human intention and corresponding to an action sequence.

However, our work focuses on the orthogonal issue, decision making, which determines the best robot arm action given some human behavior model. [17] used GMM to model human motion. While planning, the robot uses the learnt model to calculate the probability of a human occupying a voxel and that acts as a cost function during motion planning using STOMP [18]. Similar approach is used in [19] where an extra layer of task planning is added that they solved using MDP and generated an appropriate motion plan using ITOMP [20]. These approaches relied on short term prediction of human motion. Recent work [20] tackled the same problem in a factory setting and addressed the need for human predictions involving significantly longer time horizons and then planning for them accordingly. They used SIPP and re-planned online at every time step. All these methods have drawbacks mentioned in the introduction section.

POMDP planning 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 manipulator control and sensing uncertainties into decision making systematically. [22] 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. In this work, we propose to check the effectiveness of online POMDP methods in solving human robot collaboration tasks.

III. TECHNICAL APPROACH

This section describes the different technical components of our approach including our POMDP model, the DESPOT algorithm, and the use of Probabilistic Roadmaps that underpin its performance.

A. 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'.

One method for handling the lack of direct state observability is to maintain a belief over all the possible states. Let b_{t-1} be the belief at time $t-1$. If the system takes an action a_t and gets an observation z_t at the next time step t, then using Bayes' rule, we get the new belief b_t as:

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

where η is a normalization constant.

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:

$$V_\pi(b) = \mathrm{E}\big(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t)) | b_0 = b \big) \tag{2}$$

B. Problem formulation as a POMDP

Our approach utilizes a POMDP to model both the manipulator and the dynamic obstacles (humans) around it, generating control solutions that account for uncertainty in the environment.

1) State Modeling: The state vector in our dynamic environment navigation task POMDP consists of the manipulator state and a vector of dynamic obstacle states. The manipulator state consists of end effector position (x_m, y_m, z_m), and all the joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$. The state vector contains n_{human} human states whose future motion intentions are not directly observable, contributing uncertainty in the problem formulation. The state of the i^{th} human consists of its joint position $(\alpha_1^i, \alpha_2^i, \alpha_3^i, \alpha_4^i, \alpha_5^i, \alpha_6^i, \alpha_7^i)$, and its intended goal location g_i. The intention of the human is modeled as a goal location, which is hidden from the manipulator and must be inferred from its observed behavior.

2) Action Modeling: The action in the formulated POMDP consists of choosing the goal location and moving towards it for one time step, i.e., an object to go and grab on the table. The naive way to do this is by moving towards the chosen object in a straight line path. The change in robot joint angles to perform this motion can be calculated using any existing Inverse Kinematics library. However, this straight line trajectory is infeasible in the presence of static obstacles in the environment. Various motion planning algorithms [18], [20] have been developed to obtain a path for the manipulator in the presence of obstacles in the environment using optimization techniques. We propose to use a Probabilistic Roadmap or PRM for this purpose. The manipulator moves along PRM nodes and at every node the possible actions are to move to any of the adjacent neighbors. Further details on the PRM are provided in Section III-D.

3) Observation Modeling: An observation in our POMDP model is a vector consisting of the manipulator end effector position and the discretized position of the hand for all the n_{human} humans. 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 human'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.

4) Reward Modeling: The POMDP's reward model guides the manipulator towards an optimal behavior which is safe, collision-free, and reaches the goal efficiently. We considered the following rewards in our model.

- Goal Reward: If the manipulator reaches within distance D_g to the goal, then there is a large positive reward R_{goal}. This reward is modeled to encourage the manipulator to reach its goal.
- Obstacle Collision Penalty: If the manipulator reaches 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 manipulator from colliding with static obstacles.
- Human Collision Penalty: If the manipulator is moving and passes within a distance D_{hum} to a human in the environment, then there is a substantial negative reward of R_{hum}. If the manipulator is stationary, then we assume the human is responsible to avoid it. This reward is modeled to ensure safety of the human as well as the manipulator.
- Sudden Stop Penalty: If the manipulator 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.

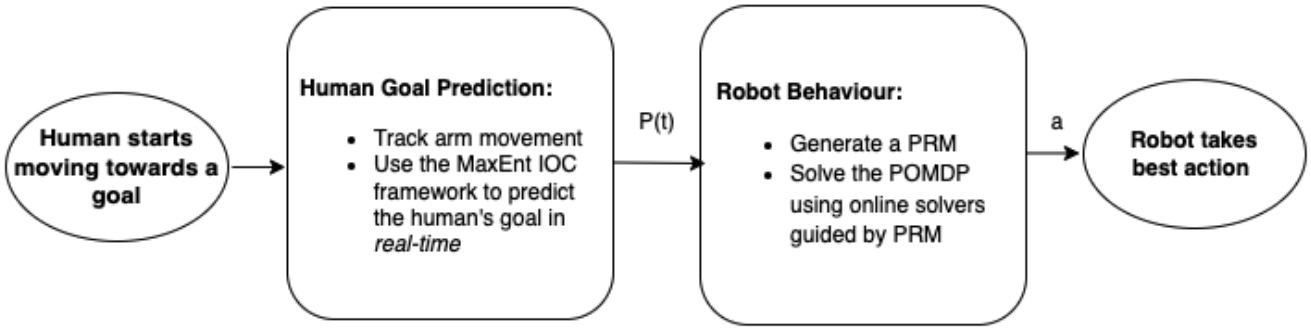


Fig. 2: Block diagram that shows different components of the proposed approach and how they collectively work.

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

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 manipulator forward by applying a for time step Δt and move all the humans towards their sampled goal location. The i^{th} human is moved towards g_i using the human model and a small random noise ω_i is added to it. The details on human model used for this work are presented in section V-A. While complex human models exist, 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.

C. Solving POMDPs Online with DESPOT

We use a state-of-the-art belief tree search algorithm, DESPOT [3] for finding a policy for our POMDP online. Its key strength is handling continuous state space and large observation spaces. To overcome the computational challenge of exploring a large belief tree, DESPOT samples a set of K "scenarios", summarizing the execution of all policies under these sampled scenarios. 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 a long horizon planning problem, it is possible that due to limited computational time, the tree search might never find the sparse positive terminal rewards that are typical in go tasks. As a result, the planner can fail to identify good actions. To prevent that, the online planner must have access to an effective roll-out policy for every belief node to obtain a lower bound on the value estimate. For our proposed formulation, the roll-out policy executes a path from the manipulator's current location to its goal aided by the use of a multi-query planner (e.g., PRM). We use a reactive controller to execute that path. If there are no humans within distance D_{far} from the manipulator, then it moves, otherwise it stays stationary. 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 manipulator is not stationary and is within distance D_{hum} from any human, then the bound is R_{hum}. Otherwise, it is $\gamma^t R_{goal}$ where t is the time taken by the manipulator to reach the goal along the chosen path assuming that the manipulator can continuously move with no dynamic obstacles (e.g., humans) 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.

D. 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 [23]. This method of motion planning is generally used for manipulators in an environment with static obstacles. For a robotic manipulator interacting in an environment, the graph nodes correspond to $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ coordinates in the joint angle space and the edges correspond to collision free linear paths between those configurations. In this work, we assigned the manipulator's starting configuration and configuration at all the goal locations for grasping as nodes in the PRM, and randomly sampled more nodes in the environment to have a total of N_{PRM} nodes. We added edges by connecting each node to its k nearest neighbors to which a collision-free straight line motion is possible. The euclidean distance between the two nodes represent the weight of the edge between them. For every node in the PRM, we find the shortest path from that node to all the possible goal configurations/nodes in the PRM. This is an offline step

and needs to be done just once for any given environment. In order to find a path from any node in the PRM to the goal configuration for the roll-out policy, we just follow the precomputed path on the PRM from that node to the goal node.

E. Tracking Belief for POMDP via Human Goal Prediction

The partially observable variables in the POMDP formulation are human 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 human goal locations for a given environment, the belief over all such intentions for each human forms a discrete probability distribution. Changes in goal can also be captured by this belief tracker.

The human is modeled as a single point $p_t \in \mathbb{R}^3$ located at the end of one of the human's hands, instead of a set of joint angles. In practice, this can be measured using a motion capture system, depth cameras, or April Tags. Given the hand's trajectory i.e. a sequence of points, we first infer a probability distribution over the human's goal $P(g^h)$ for each $g^h \in G$, where G is the set of goal positions. Let p_t be the latest point measured for the human's hand and p_0 be the hand's initial point. We model the hand's trajectory as a sequence $\xi_{p_0 \to p_t}$ and compute $P(g^h|\xi_{p_0 \to p_t})$ for each goal $g^h \in G$. Applying Bayes rule yields the equation:

$$P(g^h|\xi_{p_0 \to p_t}) = \frac{P(\xi_{p_0 \to p_t}|g^h)P(g^h)}{P(\xi_{p_0 \to p})}$$

We assume all goals and trajectories are equally likely such that $\frac{P(g^h)}{P(\xi_{p_0 \to p})} = \eta$ where $\eta \in \mathbb{R}$.

$$P(g^h|\xi_{p_0 \to p_t}) \propto P(\xi_{p_0 \to p_t}|g^h)$$

We then follow Srinivasa and Dragan in their formulation of Maximum Entropy Inverse Optimal Control (MaxEnt IOC) [25] which shows that minimizing the worst-case predictive loss results in a model where the probability decreases exponentially with the path's cost where $P(g^h|\xi_{p_0 \to p_t}) \propto e^{-C(\xi_{p_0 \to p_t}, g^h)}$.

We follow Srinivas et. al [24], [26] in defining $C(\xi_{p_0 \to p_t}, g^h)$ as the difference between the expected length of the current trajectory to the goal and a minimum length path to the goal from the initial point. We use a straight line heuristic $h(p_t, g^h) = \|g^h - p^t\|$ to compute the expected length traversed from the current position to the goal. The cost of the full path can be written as:

$$C(\xi_{p_0 \to p_t}, g^h) = \sum_{i=1}^{t} (\|x_i - x_{i-1}\|) + \|g^h - p_t\| - \|g^h - p_0\|$$

Then, at each time step, we compute $P(g^h|\xi_{p_0 \to p_t}) = \eta e^{-C(\xi_{p_0 \to p_t}, g^h)}$ for each $g^h \in G$. We then normalize these values by solving for η and pass this as a list of goal-probability pairs $(g^h, P(g^h|\xi_{p_0 \to p_t}))$ into the POMDP solver.

IV. SIMULATION ENVIRONMENT

In our simulated human-robot collaborative task, the "human" and UR5e robot must collect all cans from the table as efficiently as possible. For simplicity, we are going to assume that there is only one human in the environment. The simulated human attempts to collect the cans remaining on the table in random order in a linear fashion, without regard to the robot's actions. The robot is not aware of the human's intended goal (i.e. the object they plan to collect) at any given time; instead, it must observe the human's motion and maintain a belief over the human's intended goal, continually updating its belief distribution at each time step. The manipulator then tries to figure out the best action to execute in the environment, i.e., "can" to go towards by reasoning over the uncertainty in human's intention estimation using a POMDP framework. The task is considered complete when all objects have been removed from the table. The controller, including the inverse kinematic solver for the UR5e, along with the probabilistic roadmap were implemented in Python using the NetworkX library.

V. TECHNICAL CHALLENGES

This section describes the major technical challenges we faced while implementing this work and the smart alternatives we designed to get a working demo.

A. Human Modeling

In order to design an autonomous system that can work in the same workspace with a human safely and efficiently, there is a need to model the human and predict the behavior of the human in the environment for future time steps. Consequently, accurate mathematical models are required to model the behavior of a seven degree of freedom human arm. A lot of research efforts have focused on addressing this challenging problem. Implementing one of them to model a human arm could be a class project on its own. Given the time constraint for this project, we use a simplistic model for human and its behavior. We have implemented two different alternatives for this work. In the Bullet physics engine, we have modeled the human arm as a red spherical ball as shown in Fig.3. In the Webots simulator, we have modeled the human arm as a six degree of freedom robotic arm as shown in Fig.4.

B. POMDP Solver in Julia

Current state of the art online solvers for solving a POMDP [3], [10] are implemented in high level high performance programming language Julia. However, since Julia is still in the initial stage of gaining popularity, we don't have good open source physics engines like Bullet or simulators like Webots that support Julia. Both Bullet and Webots provide good support for other programming languages, for instance Python. As a result, the working environment can be modeled using Python while the POMDP can be solved using Julia. This requires smooth integration for exchanging information between Julia code and Python code. We tried to achieve this using different approaches. Firstly, we used the Robot

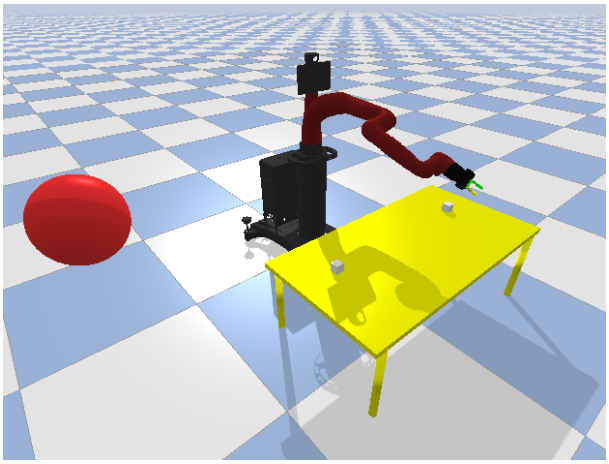


Fig. 3: Experimental setup in the Bullet physics engine. The red ball is used to model the human arm. There are two goal locations on the table and both robot arm and the human are trying to go towards different goals.

Operating System (ROS) to bridge the gap and exchange information. Although both Julia and Python have ROS support, only older versions of Julia have ROS support while the packages for solving POMDP online require newer versions of Julia. Due to this inconsistency, this approach couldn't work. Secondly, there are packages that can call Julia code from Python (PyJulia) and to call Python code from Julia (PyCall.jl). However, neither of these packages are well supported and don't work well with complicated online solvers. As a result, we couldn't use this approach either. Another potential fix is to set up APIs to request data from Julia code to Python code and vice versa. However, making these calls can introduce extra latency and that is something we can't afford in real time planning.

VI. RESULTS AND DISCUSSION

Due to difficulties in integration between Julia and POMDP code, we were unable to execute the end to end autonomous pipeline that we wished to execute. However, to not get stuck and develop a system for autonomous robotic manipulation among dynamic obstacles, we used a PRM based approach that is similar to the "predict then act" controller [24].

For the Bullet physics engine, we have assumed that there are two possible goal locations on the table. The human arm which is modeled as a ball moves in a straight line path from a randomly sampled point in the environment to one of the goals on the table. The entire trajectory of the ball is assumed to be a static obstacle. The PRM is generated by sampling points in the (x, y, z) coordinates that don't collide with the obstacle. We then use the A* algorithm to find the shortest path for the manipulator from its current state to the other goal location on the table. The path is implemented using an existing Inverse Kinematics library. The demo of one such trajectory is attached with our submission.

For Webots, the simulation environment has been described in Section IV. The human arm is modeled as another UR5e robotic arm. The PRM is generated by sampling points $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ in the joint angle space. Since we know the model for the human arm and its intended goal, we can generate the arm's trajectory. We reject a sampled PRM node if it leads to a collision with the can locations or the table in the environment or the human arm's trajectory. We keep sampling nodes until the PRM has N_{PRM} nodes. We then use Dijkstra's algorithm to find the shortest path for the manipulator from its current state to one of the can locations on the table that the human arm is not going towards. Since the PRM nodes are in the joint space, the path can be implemented by just rotating the manipulator joints. Figure 4 shows a two-robot table clearing scenario when the right robot uses the PRM-based approach. In addition, a demo is attached with our submission.

Our demonstration videos show that the approach of using a PRM to generate trajectories for a manipulator is effective in obtaining efficient behavior and avoiding collisions with dynamic agents in the environment. We expect to see similar results when we overcome the integration issues to set up the entire pipeline and merge it with a POMDP planner.

VII. FUTURE WORK

There are multiple possible extensions for our work in the future. At present, the assumed human model is pretty naive and not realistic. We wish to explore state of the art methods for modeling human behaviour and use that for our work and if possible, even improve it. Also, due to issues in Julia and Python integration, we couldn't build an end to end pipeline. We wish to focus on fixing that either by exploring more alternatives or by reinventing the wheel and implementing the state of art online POMDP solver, DESPOT in Python. We plan to work on this project in the summer to create an end to end novel system, and test its functionality on a 7-DOF robot arm in the real world. We aim to run a user study, see the performance of our proposed system in comparison to prior state of the art approaches and submit our work and results as a conference paper in HRI 2023.

ACKNOWLEDGMENT

The students would like to thank Dr. Nikolaus Correll for his constant guidance and support throughout this course.

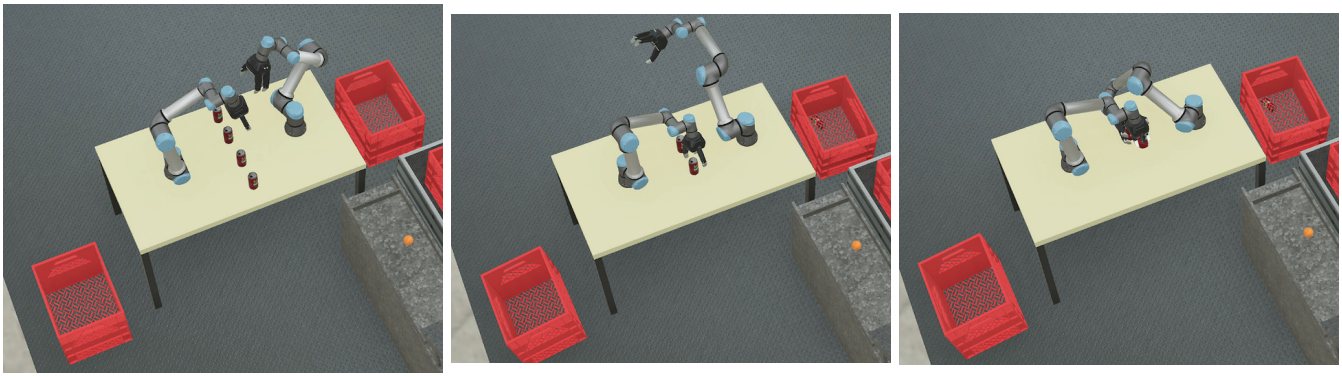
REFERENCES

[1] Cassandra, Anthony R. "A survey of POMDP applications." Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes. Vol. 1724. 1998.

[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] 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.



(a) (b) (c)

Fig. 4: The left UR5e robot executes a fixed trajectory while the right UR5e robot implements a PRM-based approach. (a) The right robot detects a possible collision and (b) takes an alternative path found in the PRM. (c) Both robots grasp the cans without collision with each other.

[5] 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.

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

[10] 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.

[11] 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

[12] 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.

[13] 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

[14] 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).

[15] Y. Li and S. S. Ge, "Human–Robot Collaboration Based on Motion Intention Estimation," in IEEE/ASME Transactions on Mechatronics, vol. 19, no. 3, pp. 1007-1014, June 2014.

[16] M. Awais and D. Henrich, "Human-robot collaboration by intention recognition using probabilistic state machines," 19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010), Budapest, 2010, pp. 75-80.

[17] Mainprice, Jim, and Dmitry Berenson. "Human-robot collaborative manipulation planning using early prediction of human motion." 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013.

[18] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in ICRA, 2011.

[19] Park, Jae Sung, Chonhyon Park, and Dinesh Manocha. "Intention-Aware Motion Planning Using Learning Based Human Motion Prediction." Robotics: Science and Systems. 2017.

[20] Park, Chonhyon, Jia Pan, and Dinesh Manocha. "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments." Twenty-Second International Conference on Automated Planning and Scheduling. 2012.

[21] Unhelkar, Vaibhav V., et al. "Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time." IEEE Robotics and Automation Letters 3.3 (2018): 2394-2401.

[22] 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.

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

[24] S. Javdani, S. Srinivasa, and J. A. D. Bagnell, "Shared Autonomy via Hindsight Optimization," Robotics: Science and Systems, 2015

[25] Dragan and S. Srinivasa, "A policy-blending formalism for shared control," The International Journal of Robotics Research, vol. 32, no. 7, pp. 790805, 2013.

[26] Pellegrinelli, Stefania, et al. "Human-robot shared workspace collaboration via hindsight optimization." 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016.