Planning with Approximate Preferences and its Application to Disambiguating Human Intentions in Navigation

Bradford Neuman[†] Maxim Likhachev[†]

Abstract— **This paper addresses the problem of planning in the presence of humans modeled as dynamic obstacles with multiple hypotheses on their trajectories and actions which can disambiguate between the hypotheses. To solve this problem, we develop and analyze a generalization to the PPCP (Probabilistic Planning with Clear Preferences) algorithm that allows us to efficiently solve problems with approximate preferences on missing information. The approach finds policies with bounded suboptimal expected cost and scales well with the number of people, only disambiguating between the trajectories of people when necessary. We present simulated results as well as experiments on two different physical robots demonstrating the capability of this planner.**

I. Introduction

Recently, the robotics community has made great progress toward realizing capable autonomous systems. With the Urban Challenge vehicles and Google Chauffeur autonomous driving project, we have seen systems that can react to many real-life situations on the road, including obeying traffic rules and avoiding other moving vehicles. These systems are impressive for many reasons, not the least of which is that they can handle planning in dynamic environments. Systems like BOSS use the structure in the environment and sensor fusion to model the most likely trajectory of other vehicles, and then plan around them with a margin of safety [1].

In many situations, it may not be possible to identify a single trajectory for each human in the environment. At an intersection, for example, people's actions become uncertain. State of the art tracking techniques can produce better results by creating a probabilistic model with several possible trajectories [2]. Another approach uses inverse optimal control to model humans walking through an environment, and then plans paths that have a high cost if they interfere with the human [3]. Other successful approaches include using a mixture of Gaussian Processes [4], or a Hidden Markov Model to learn uncertain dynamic trajectories from prior data [5].

For dynamic obstacle planners to advance further, we need to take uncertainty in the intentions of humans into account. Ideally, the robot would use additional sensors or computational resources to disambiguate between each possible trajectory for each person and make the problem deterministic, but this is not generally possible. The next best thing would be to optimally focus our resources to disambiguate the hypotheses of the people that matter most to the robot's navigation task by planning sensing actions, such as aiming a pan/tilt camera, or simply driving closer to take better readings. Finally, we could directly disambiguate intentions vocally.

† Robotics Institute Carnegie Mellon University Pittsburgh, PA 15213 {bneuman,maxim}@cs.cmu.edu

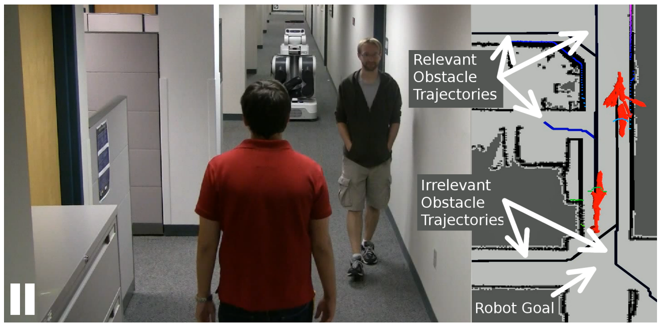

Relevant Obstacle Trajectories
Irrelevant Obstacle Trajectories
Robot Goal

Fig. 1. The PR2 robot traverses an environment with humans that have multiple trajectory hypotheses. The planner decides which people are important, and disambiguates their intentions by announcing its presence and observing the result.

This paper formulates this problem and shows how it can be solved with a principled decision theoretic approach called Probabilistic Planning with Clear Preferences (PPCP). We introduce and analyze a generalization to the algorithm, which relaxes a strict assumption of PPCP and has applications outside this domain. We present an approach that automatically disambiguates between trajectories of people that may impede the navigation of the robot.

In general, this type of planning problem would fall into the category of planning with incomplete information, and more generally falls into a broader class of planning for Partially Observable Markov Decision Processes (POMDPs) [6]. Planning optimally for POMDPs, in general, and planning with incomplete information and with sensing, in particular, are known to be intractable [7], [8]. The original PPCP algorithm avoids this issue by assuming that we can identify a priori which outcomes of uncertain actions are known to be the best [9]. This paper introduces a way to handle cases when we do not know for certain, but have a bounded approximation on which outcomes we think may be the best. We analyze the modified algorithm in this new context, and show that it is tractable both in simulation and on real robotic platforms. In simulation, we show that this technique can solve problems with 10 people (uncertain dynamic obstacles) and over 20 trillion possible belief states in a matter of seconds.

II. Related Work

Recently, planning techniques have been developed that deal with deterministic dynamic obstacles very efficiently [10]. Other techniques can model static and changing portions of a map very efficiently and globally plan with respect to the static obstacles while dealing with dynamic obstacles locally [11]. Another approach is to treat planning with dynamic obstacles as a special case of multi-agent or cooperative path-finding [12]. Another technique takes into account uncertainty in the obstacles and can perform obstacle avoidance by limiting the planner to computed safe zones and velocities [13]. There has been much prior work in optimizing sensing and active vision, where the goal of the system is to maximize performance on a perception task such as object classification [14].

Recent techniques have combined object detection with navigation, optimizing the cost of a mobile robot trajectory that has explicit value in sensing objects [15]. Other work has applied a modified RRT to a Gaussian Process model of dynamic uncertain obstacles and has demonstrated effective use on a robotic platform [4]. Another approach uses A* on an occupancy grid and demonstrates results on a robot which are better than using a simple linear velocity model [5].

None of these approaches, however, reason about uncertainty in the intentions of humans by planning to disambiguate these intentions when needed, and none of them produce plans that are optimal in expectation or have upper bounds on suboptimality.

III. Application Domain

We wish to have a robot autonomously navigate in a dynamic environment containing humans. We model humans as uncertain agents that are dynamic obstacles the robot must avoid. To navigate effectively and avoid collisions with these people, the robot needs to be able to avoid their future locations. Unfortunately, it is not possible to predict exact future trajectories. Often, the robot can disambiguate between the possible future trajectories by asking the involved person which way they are going, honking a horn to get the persons attention, or simply scheduling future sensor readings and replanning when more information will be available. We call these *focus* actions. The challenge for the planner is to decide when and how to use these actions since they depend on both the position of the robot and the possible positions of the people in the environment (see section Sec. VI-A for examples).

In our representation, each human d_i has a set of possible trajectories $\{\tau_i^0, \ldots, \tau_i^{n_i-1}\}$. There is a hidden state h_i which corresponds to the actual trajectory d_i follows. h_i can be an integer j, in which case the robot knows that d_i follows τ_i^j, or $h_i = u$, in which case the trajectory is unknown to the robot, but we have a probability distribution over its n_i possible values. In order to determine the value of the hidden state, the robot can execute its *focus* action on person i if the person is visible and within range. This will cause h_i to resolve to an integer in $\{0, \ldots, n_i - 1\}$.

We assume that people will follow exactly one of the possible trajectories, and that there is no noise in sensing or actuation, but these issues can be dealt with on real systems by replanning and adding padding (both temporally and spatially) to the dynamic obstacles, as explained in Section Sec. V and demonstrated in section Sec. VI-A.

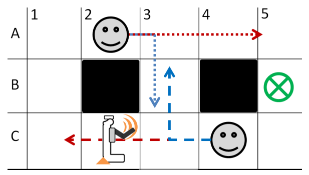


Fig. 2. An example environment with two humans. The belief state of the robot is $X = \{R = C2, t = 1, H = [u, u]\}$.

We will be planning in the space of belief states, $X = [S(X); H(X)]$, which include the current deterministic state of the robot, $S(X)$, and a set of hidden variables, $H(X)$, which encode the current probability distribution over the unknown variables. $S(X)$ could be, for example, (R, t), the position of the robot and current time. In the environment in Fig. 2, both people are moving along unknown trajectories as shown in the red and blue lines. After executing an action such as moving to the west one cell, there is only one resulting state, $X' = \{R = C1, t = 3, H = [u, u]\}$ (assuming the move took two timesteps). Notice that H does not change. Now, assume that person 0 (the one in cell C4 in Fig. 2) is visible from X, so the robot can focus on it. After executing the focus action, there are two possible resulting states:

$$X'_0 = \{R = C2, t = 6, H = [0, u]\}$$
$$X'_1 = \{R = C2, t = 6, H = [1, u]\}$$

In both states, the deterministic part is the same (the robot has not moved and it spent 5 timesteps focusing), but the value of h_0 has changed. In X'_0, the robot knows that person 0 will follow τ_0^0 (the blue dashed path), while in X'_1, person 0 will follow τ_0^1 (the red dashed path). In general, the deterministic part can differ as well.

IV. Algorithm

A. Clear Preferences

While in general, decision-theoretic planning that takes uncertainty about the environment into account is computationally difficult, it turns out that many such problems exhibit a special property: one can clearly identify beforehand the best (called *clearly preferred*) values for the variables that represent the unknowns in the environment [9].

Clear preferences can be defined mathematically as follows. The clearly preferred value b ("best") of an unknown variable h is such a value that for any belief state X and action a that senses (directly or indirectly) the value of h, there exists a successor belief state X' at which the value of h is known to be the clearly preferred value ($h = b$) and:

$$X' = \underset{Y \in \text{succ}(X,a)}{\operatorname{argmin}} \; c(X, a, Y) + v^*(Y) \qquad (1)$$

where $c(X, a, Y)$ is the cost of executing action a at state X and ending up at state Y, and $v^*(Y)$ is the expected cost of executing an optimal policy from the belief state Y.

For example, in the problem of an indoor robot navigating with uncertainty about doors being open or closed, a state X might represent the robot when it is about to sense if a door is open or closed. The sensing operation will have two outcomes, the preferred outcome, X', in which the door is open and an alternate outcome in which the door is closed. Since the actions the robot can take if the door is closed are a subset of the actions for an open door, the open door outcome is *clearly preferred*.

Algorithm 1 Planning with Approximate Preferences. v values are initialized to an admissible estimate of v^*. S_i in COMPUTEPATH is shorthand for $S(X_i)$, the deterministic part of the state X_i.

```
1: **procedure** COMPUTEPATH($X_\mathrm{p}$)
2:    $\forall S\ g(S) \leftarrow \infty, \quad \underline{\mathrm{OPEN}} \leftarrow \emptyset$
3:    $g(S_\mathrm{goal}) \leftarrow 0, \quad \mathrm{bestA}(\mathrm{S_{goal}}) \leftarrow$ **null**
4:    insert $S_\mathrm{goal}$ into $\underline{\mathrm{OPEN}}$
5:    **while** $g(S_\mathrm{p}) > \min_{S \in \underline{\mathrm{OPEN}}} g(S) + heur(S_\mathrm{p}, S)$ **do**
6:       remove $S$ with min $g(S) + heur(S_\mathrm{p}, S)$ from $\underline{\mathrm{OPEN}}$
7:       **for all** $a$ and $S'$ s.t. $S = \mathrm{succ}_{X^u_\mathrm{p}}(S', a)^{b'}$ **do**
8:          Compute $Q(S', a)$ according to Eq. 3
9:          **if** $g(S') > Q(S', a)$ **then**
10:            $g(S') \leftarrow Q(S', a)$
11:            $\mathrm{bestA}(\mathrm{S}') \leftarrow \mathrm{a}$
12:            (re-)insert $S'$ into $\underline{\mathrm{OPEN}}$with $g(S') + heur(S_\mathrm{p}, S')$
13:         **end if**
14:      **end for**
15:   **end while**
16: **end procedure**
17: **procedure** UPDATEMDP($X_\mathrm{p}$)
18:   $X \leftarrow X_\mathrm{p}$
19:   **while** $S(X) \neq S_\mathrm{goal}$ **do**
20:      $v(X) \leftarrow \max(v(X), g(S(X))$
21:      $v(X^u) \leftarrow \max(v(X^u), g(S(X^u))$      // $S(X) = S(X^u)$
22:      $\pi(X) \leftarrow \mathrm{bestA}(\mathrm{S(X)})$
23:      $X \leftarrow \mathrm{succ}(X, \pi(X))^{b'}$
24:   **end while**
25: **end procedure**
26: **procedure** MAIN($X_\mathrm{start}$)
27:   $X_\mathrm{p} \leftarrow X_\mathrm{start}$
28:   **while** $X_\mathrm{p} \neq \emptyset$ **do**
29:      COMPUTEPATH ($X_\mathrm{p}$)
30:      UPDATEMDP ($X_\mathrm{p}$)
31:      $X_\mathrm{p} \leftarrow$ FINDPIVOT ($X_\mathrm{start}$)      // see Sec. IV-B
32:   **end while**
33: **end procedure**
```

B. Approximate Preferences

While many problems exhibit clear preferences, there are many others for which it is difficult or impossible to predict clear preferences. However, in many of these cases we can come up with *approximate preferences*. In particular, for our domain of interest, one way we could come up with these approximate preferences would be to plan a path assuming no unknown obstacles exist at all. If one of the hypothesized trajectories of a human invalidates the shortest path to the goal for the robot, chances are it is a worse trajectory than another hypothesis. There are situations, however, where this approach can predict the wrong outcome, such as when a person approaching the robot actually clears a shortcut that saves the robot time overall.

Mathematically, we define an approximate preference, b' as follows. For all states X and available actions a, it holds that:

$$c(X, a, \mathrm{succ}(X, a)^{b'}) + v^*(\mathrm{succ}(X, a)^{b'}) \leq \alpha\big(c(X, a, \mathrm{succ}(X, a)^d) + v^*(\mathrm{succ}(X, a)^d)\big)\ \forall d \quad (2)$$

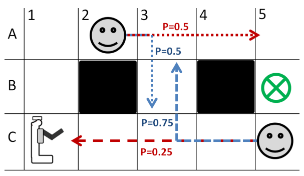
Here, the notation $\mathrm{succ}(X, a)^d$ refers to the single successor of state X after executing action a with the outcome d, meaning $h = d$ (we assume at most one h per stochastic transition). This equation says that for a given state and action, the policy's expected cost with the approximate preference b' is no worse than α times what it would be for another outcome (with $\alpha \geq 1$). If we have clear preferences, as in the office door problem, $\alpha = 1$ because we already have the best preference and Eq. 2 reduces to Eq. 1.

To efficiently plan under approximate preferences, we apply the generalized PPCP algorithm shown in Algorithm 1. This algorithm is a slight modification from the main PPCP algorithm as presented in [9]. In the case of clear preferences, the algorithm presented here behaves equivalently to the original PPCP algorithm, and therefore has the same properties of completeness and optimality of the resulting policy under the conditions given in [9].

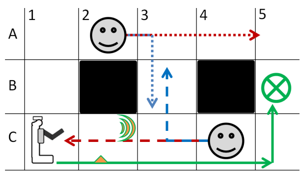
The PPCP algorithm consists of four procedures. COMPUTEPATH performs a single A* like search from the *pivot* state, then UPDATEMDP takes the computed path into account, updating the MDP variables. FINDPIVOT searches the current partial policy for the next pivot. The MAIN procedure runs each of these procedures until there is no valid pivot (the algorithm has converged). For use on a real robotic platform, the MAIN procedure returns partial policies for the robot to begin to execute without reaching full convergence. Note that the original PPCP presentation included FINDPIVOT in the MAIN procedure, but for simplicity it is separated out and briefly explained later in this section.

It is easiest to understand this algorithm by example, so we will consider the environment shown in Fig. 3(a). The robot starts in cell C1 and is trying to reach B5 by moving in one of the 4 cardinal directions, or waiting in place. Collisions with a dynamic or static obstacle are not allowed. There are two humans. The first starts in cell C5 and moves at the same speed as the robot (trajectories shown in dashed lines). There are two predicted future trajectories for this person. The first (blue) moves north with probability 0.75, while the other (red) moves straight west. The robot selects the blue trajectory as approximately preferred. The other person starts in cell A2 and moves more slowly, only one cell every two timesteps (dotted line). It has two trajectories with equal probability, the (approximately) preferred shown in blue that turns south down the hallway, and the non-preferred that moves straight east.

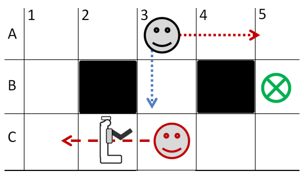
The first step is to run COMPUTEPATH to compute a path from the start state to the goal with both unknown variables set to unknown (Fig. 3(b)). The COMPUTEPATH



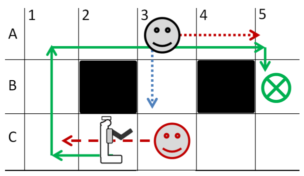
(a) Start state:
$R = C1, t = 0, H = [u, u]$



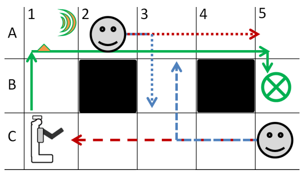
(b) First policy



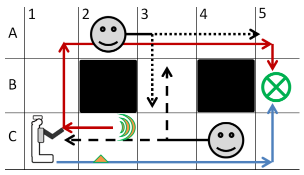
(c) Next pivot state:
$R = C2, t = 2, H = [1, u]$



(d) ComputePath result



(e) Final ComputePath call:
$R = C1, t = 0, H = [u, u]$



(f) Final Policy.

Fig. 3. An example run of PPCP showing 3 iterations. The final policy is to follow (b) in the preferred case and (d) otherwise.

procedure is very similar to a backwards A* search. Just like A*, it maintains g values (initially set to infinity) and an open list that is a priority queue sorted by $g + heur$, using an admissible heuristic function. On Line 7, states that are predecessors of the current state are generated. The successor function is of the form $\text{succ}_Z(S, a)^{b'}$, which corresponds to the preferred successor of taking action a from the state $[S; H(Z)]$ (the deterministic state S with hidden variables set according to Z). In Line 7, the hidden variables are set according to X_{p}^{u}, which corresponds to $H(x_{\text{p}})$ with all $h_i = b$ replaced with $h_i = u$. This means that we are "forgetting" the preferred outcomes that are known in the belief state X_{p} and pretending they are unknown. This allows PPCP to find a full policy by running a series of deterministic graph searches.

In the example in Fig. 3, most transitions are deterministic (only one successor). During the first ComputePath search the only non-deterministic expansion is the state shown in Fig. 2, where the orange arcs represent a sensing action.

An important difference between ComputePath and standard A* is the way Q is computed. The standard definition of $Q(Y, a)$ is the expected value of taking action a from state Y and then executing the current policy from the resulting state. As the algorithm is running, we maintain v as an estimate of the expected value of the optimal policy. Therefore, we could use the following equation:

$$Q(Y, a) = \sum_{Z \in succ(Y,a)} P(Y, a, Z) \cdot (c(S(Y), a, S(Z)) + v(Z))$$

Since we have a notion of approximate preferences, we tend to believe that the preferred outcome will have lower cost, so we replace this equation with:

$$Q(Y, a) = \sum_{Z \in succ(Y,a)} P(Y, a, Z) \cdot \max \begin{pmatrix} c(S(Y), a, S(Y')) + v(Y'), \\ c(S(Y), a, S(Z)) + v(Z) \end{pmatrix}$$

where Y' is the preferred outcome. In the case of true clear preferences (satisfying Eq. 1), these equations are equivalent. Under approximate preferences, however, we may now be over-estimating the cost (by at most α for each branch).

Since ComputePath searches backwards from the goal, it already has an estimate, g, for the preferred outcome. This estimate is more recent than the v value that may have been set in a previous iteration of PPCP, so the final equation is:

$$Q(Y, a) = \sum_{Z \in succ(Y,a)} P(Y, a, Z) \cdot \max \begin{pmatrix} c(S(Y), a, S(Y')) + g(Y'), \\ c(S(Y), a, S(Z)) + v(Z) \end{pmatrix} \tag{3}$$

When action a executed at Y is deterministic, this equation reduces exactly to the equation used in normal A* search. In our example, the Q value is computed at the state shown in Fig. 2. If the person follows the blue path, we have a valid g value already. Otherwise, we use the current estimate of v associated with the state shown in Fig. 3(c).

We assume that there either exists a finite cost policy, or there is no solution. This means that for any given pivot state, ComputePath can terminate and find a valid path from X_{p} to the goal, or report that there is no solution. This path is the minimum cost path according to the Q values defined in Eq. 3. Fig. 3(b) shows the path found in the first ComputePath iteration. The path drives east, senses the person, and then continues to the goal (assuming the person moves north).

In the UpdateMDP function, PPCP takes this new path into account and updates the v values and the policies along it. Because the successor function in Line 7 uses X_p^u, we can also update the $v(X^u)$ value on Line 21. Line 20 and Line 21 are modified from the original PPCP algorithm to handle approximate preferences. Under clear preferences, the addition of the $\max$ operator does not change the behavior because the v values are guaranteed to increase monotonically [16]. However, under approximate preferences, a previous iteration of ComputePath could have used a Q value that was too high and could potentially have found a lower value during this iteration. To assure constant progress of the algorithm, we do not allow the v values to decrease, which guarantees termination because v is upper bounded and grows by an amount lower bounded by a small positive constant. This turns out to be the same update rule as listed in Section 6.3 of [9] where an optimization to improve the estimates of v-values is explained, although the analysis of the algorithm presented here (under the new assumption of approximate preferences) and the domain are novel.

Fig. 3(b) shows the policy after the first iteration of COMPUTEPATH . It is incomplete because it does not define a policy where the sensing action arrives at the non-preferred outcome where person 0 continues to move west.

The last section of the MAIN procedure calls FINDPIVOT to find the next pivot state. The FINDPIVOT procedure returns any non-goal state X that can be reached by following the policy π starting from X_{start} and is either a state for which no policy is defined or satisfies:

$$v(X) < \sum_{Y \in \text{succ}(X,\pi(X))} P(X,\pi(X),Y) \cdot \big(c(S(X),\pi(X),S(Y)) + v(Y)\big) \tag{4}$$

This equation says that the v value of the state must be less than what it should be according to the expected value over all outcomes reachable by following the policy actions. To correct the v value of this state, we need to run another COMPUTEPATH iteration. If no such state X can be found, FINDPIVOT must return $\emptyset$, which will trigger termination of the PPCP algorithm, and ensure that all states along the policy have consistent v values. In our implementation, FINDPIVOT returns a state that has the maximum probability of being reached among all valid states to return.

In our example, the state shown in Fig. 3(c) is selected as the pivot because it does not have a defined policy, but is reachable along the current policy from the start. In the next iteration of PPCP, we run COMPUTEPATH to find a path from the new pivot to the robot's goal. According to the pivot shown in Fig. 3(c), the robot knows that person 0 is moving along the red trajectory straight west. COMPUTEPATH computes the path shown in Fig. 3(d) which backs up to get out of the way of person 0 and then takes the northern path to the goal. During UPDATEMDP the algorithm makes the v-value of $X = \{R = C2, t = 1, H = [u,u]\}$ smaller than what it should be according to the expected value of its policy successors. Therefore, Eq. 4 is satisfied and this state can be selected as the next pivot. PPCP now explores alternative paths and updates the v-values and policy as it goes (Fig. 3(e)). After 3 iterations, PPCP converges to a final policy shown in Fig. 3(f), where the robot drives east and senses person 0, following the blue path if the person takes the preferred trajectory. Otherwise, if the robot senses the person continuing west along the non-preferred trajectory, the robot follows the red path shown in Fig. 3(f).

C. Bounded Approximate Preferences

In this section, we derive a bound on the sub-optimality of the computed policy as a function of α, the bound on the approximately preferred outcome. First we derive α for the approximate preferences used in our domain, then we show how this leads to a general sub-optimality bound on the policy itself. We define the following notation:

$$V^d(X,a) = c(X,a,\text{succ}(X,a)^d) + v^*(\text{succ}(X,a)^d)$$

Starting with Eq. 2 using this notation:

$$\frac{V^{b'}(X,a)}{V^d(X,a)} \le \alpha \quad \forall X,a,d$$

$$\alpha = \max_{X,a,d} \frac{V^{b'}(X,a)}{V^d(X,a)} \tag{5}$$

In our problem of navigation in dynamic environments with uncertain human intentions, we can derive a bound on α by looking at the ratio between the cost of the navigation in a pessimistic world to that of an optimistic world. Because Eq. 2 is defined over all states X, we need to check this ratio at every state. However, only some states have stochastic transitions, and for deterministic transitions, preferences do not matter. To compute this bound, we run a deterministic Dijkstra's search from the goal assuming all humans follow all of their possible trajectories simultaneously, giving us the cost $A(X)$ (the search is run only over the deterministic part $S(X)$). We also run a search from the goal assuming there are no people (no dynamic obstacles) at all, giving us a cost $B(X)$. We then run Dijkstra's from the starting position of the robot, giving us $R(X)$. We use $R(X)$ to determine the places where a stochastic transition can happen. A state X with timestep t can have a stochastic transition only if a person is visible from X and $R(X) \le t$. This means that any state the robot can get to within t timesteps that has a visible person at that time is a state where the focus action could be performed and the policy could branch. We call this set of possible stochastic transitions T. Our bound is then:

$$\tilde{\alpha} = \max_{X \in T} \frac{A(X)}{B(X)} \tag{6}$$

This bound holds because any $X \notin T$ has no stochastic transitions, so preferences do not matter ($\alpha = 1$ for those states). $A(X) \ge V^{b'}(X,a)$ since policies in the world where all trajectories are simultaneously followed can be no better than policies with any single set of preferences. Similarly, $B(X) \le \min_d V^d(X,a)$ because B represents an empty world that must be at least as good or better than any valid preference. Thus, $\tilde{\alpha} \ge \alpha$ (α is the minimum bound; $\tilde{\alpha}$ is our upper bound on α).

Given the bound for each preference, the policy found by PPCP will be at most α^k times the optimal expected cost, where k is the maximum number of stochastic actions along any branch of the policy, and is known. This can be thought of as the depth of a tree where each node in the tree represents a state in the policy where a focus action takes place, and its children represent outcomes.

If there are no stochastic transitions, PPCP returns the optimal policy since there can be no errors in preference ($k = 0$). Otherwise, let $v_j(X)$ be the v value of any state X that has exactly j stochastic actions on the policy π between X_{start} and X where $0 \le j \le k$.

We proceed by induction on j. For $j = k$, there are no stochastic actions left, so $v_k(X) \le \alpha^{k-k} \cdot v^*(X) = v^*(X)$, the expected cost of the optimal policy from state X. Assume $v_j(X) \leq \alpha^{k-j} \cdot v^*(X)$ for some j. Since v is set according to Q values along the COMPUTEPATH paths, according to Eq. 3, for a state at depth $j-1$:

$$\begin{aligned} v_{j-1}(X) = & \sum_{Z \in succ(X,a)} P(Y,a,Z) \cdot \\ & \max \begin{pmatrix} c(S(Y),a,S(Y')) + g_{old}(Y'), \\ c(S(Y),a,S(Z)) + v_{old}(Z) \end{pmatrix} \\ \leq & \sum_{Z \in succ(X,a)} P(Y,a,Z) \cdot \qquad (7) \\ & \max \begin{pmatrix} c(S(Y),a,S(Y')) + g_{old}(Y'), \\ c(S(Y),a,S(Z)) + v_j(Z) \end{pmatrix} \qquad (8) \end{aligned}$$

$$\begin{aligned} v_{j-1}(X) \leq & \sum_{Z \in succ(X,a)} P(Y,a,Z) \cdot \qquad (9) \\ & \max \begin{pmatrix} c(S(Y),a,S(Y')) + g_{old}(Y'), \\ \alpha \cdot \left(c(S(Y),a,S(Z^*)) + v_j(Z^*)\right) \end{pmatrix} \\ \leq & \alpha \cdot \left(c(S(Y),a,S(Z^*)) + v_j(Z^*)\right) \qquad (10) \\ \leq & \alpha \cdot \left(c(S(Y),a,S(Z^*)) + \alpha^{k-j} \cdot v^*(Z^*)\right) \\ \leq & \alpha^{k-(j-1)} \cdot \left(c(S(Y),a,S(Z^*)) + v^*(Z^*)\right) \\ \leq & \alpha^{k-(j-1)} \cdot v^*(X) \end{aligned}$$

Eq. 7 holds because v values are monotonically increasing, so the older v value must be less than or equal to the newer one. Eq. 9 holds by Eq. 2, where Z^* is the best possible outcome. Eq. 10 follows because the second term of the $\max$ is always larger than (or equal to) the first. We have proven that the factor on α can increase by at most one per level and $v_j(X) \leq \alpha^{k-j} v^*(X)$ for all $0 \leq j \leq k$. By taking $j = 0$, we prove that $v(X_{\text{start}}) \leq \alpha^k v^*(X_{\text{start}})$. Proofs that the expected cost of $\pi(X_{start})$ is $v(X_{start})$ and that the policy is sound follow the same proof as presented for the original PPCP algorithm [16].

V. IMPLEMENTATION

To implement this planner in our domain, we define the deterministic state space of the problem as $S(X) = (x, y, t, f)$ where $f \in \{\emptyset, 0, 1, ...\}$ represents the robot's current focus, with $\emptyset$ meaning there is no focus. The policy may have many focus actions targeting many different people, but only one person is in focus at any given time corresponding to the setting of f, with $f = \emptyset$ meaning no person is in focus. The goal for the robot is only specified as a position (x, y), so we create a window of possible ending times that covers the minimum and maximum time it could take for the robot to reach the goal. The heuristic is a 2D Dijkstra's search in a world with no people.

We implemented this planner on both the segway RMP based differential-drive robot shown in Fig. 4 and the Willow Garage PR2 robot in Fig. 1. Both robots are equipped with a lidar scanner for static obstacle detection and human tracking. For our experiments, we also equipped the robots to play sound so they could vocally interact with humans. Both run ROS, Willow Garage's Robot Operating System, which allows us to integrate a number of off-the-shelf components written by people across the robotics community, including a local planner that can follow the rough trajectories generated by this algorithm.

In order to run this planner, we need a multi-hypothesis tracking system. This paper focuses on the planner, so the tracking systems we used will not be explained in detail. We implemented a simple tracker that clusters lidar points and tracks clusters that may be moving. We use a pre-defined static map that is annotated with intersections. Alternatively, the paths and probabilities could be learned automatically from recorded data, such as in [5]. For the segbot experiments, we assume the people are moving in a straight line and may or may not pause before entering an intersection. For the PR2 experiments, we also annotate the map with a set of possible goals and a probability that a random person is traveling to that goal. Using this information, we plan a path from the person to each goal and rule out any goals that the person is moving away from. This tracker limited the speeds the people in the experiments could move towards the robot, but it still allowed us to verify our approach experimentally.

In order to make the system run online, it was not possible to always allow PPCP run to full convergence. Full convergence is rarely needed, as after the first few iterations on a problem with one or two humans, the policy tends to be the same as after the algorithm converges (see Sec. VI-B). As long as at least one iteration of MAIN procedure has completed, there will be a policy at least up to the goal or first branch. This is sufficient, since the robot can improve its plan as it drives, or replan from scratch when it reaches the focus point and more information is available.

For this paper we focus on indoor environments with humans navigating the hallways. In our implementation, the robot approximates that trajectories that pause or duck out of the way of a main hallway are preferred, although this preference could be estimated online if needed.

VI. EXPERIMENTAL RESULTS

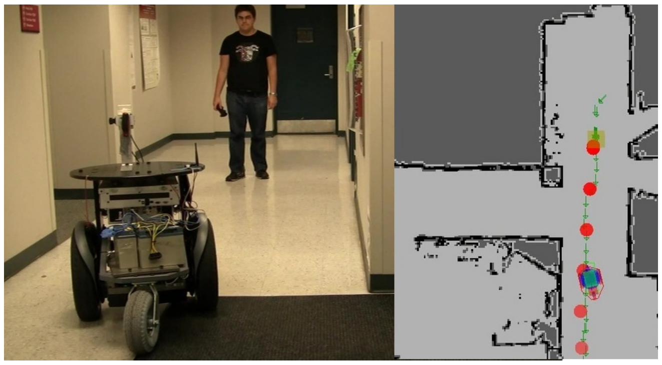


Fig. 4. The segbot plans to collect additional sensor data to determine if the person will continue walking or pause to allow the robot to pass.

A. *Robot Results*

There are two experiments presented, each with a different focusing method. Each experiment has two cases corresponding to different outcomes of a focus action, each of which can be seen in the attached video. Both methods of focusing have their own visibility function, which determines at which states they can differentiate between possible hypotheses. In this work, the focus actions were separately tested, but both actions could be included and the planner could be allowed to chose between them as appropriate.

Speech-based focus: The first focus method is for the robot to speak to the person in an attempt to resolve uncertainty in the intention. In our example (Fig. 1), the PR2 is trying to navigate to a position behind two walking people. By saying "excuse me" out loud, the robot gets the attention of the person it is speaking to, and that person either moves out of the way for the robot (toward the temporary goal just off the corridor), or continues to one of his original goals. With the PR2's sensors, we were able to detect and predict trajectories for two moving people. One person is walking towards the robot and may collide with it, while the other person is walking away from the robot. The planner is given 1.5 seconds to plan and automatically ignores the person who is moving away while it focuses to disambiguate the intention of the person approaching the robot.

Sensing-based focus: The second focus method is a simple operation where the robot stops and waits while it gathers more sensor information to determine which trajectory the person is actually following. At this point, additional obstacles or trajectories may be introduced, but the uncertainty present in the initial policy with respect to the person focused on will be gone. The visibility function for this focus action requires the robot to be within a sensing range and the hypotheses about the person to have diverged (the trajectories do not all occupy nearby (x, y) positions).

In the corresponding segbot experiment (Fig. 4), the robot is at a T–intersection and the person is approaching. The person will either pause before the intersection, or continue to walk straight towards the robot. If the person walks straight, the robot must back up to the area near the doors and move to the side to get out of the way, while if the person pauses, the robot can go in front of the person. The robot moves forward slightly to get into the "visible" range of the person, then the robot replans and at this time the person has either moved past the intersection, in which case the robot turns to get out of the way, or it has stopped before the intersection, in which case the robot moves in front of the person and proceeds along the hallway. This experiment demonstrates that the robot has successfully taken this uncertainty into account. In this experiment, the planner was given 2 seconds.

B. *Simulation Results*

We created random 100 x 100 cell ($^1/_3$ m cell width) indoor maps with randomly generated hallways and rooms, like the one shown in Fig. 5. In these experiments, people were randomly inserted at a point in the free space of the map, and 4 goals were randomly chosen which the person would plan a path to with a time resolution of 0.5 seconds. We experimented with 1 through 10 people and generated 450 random dynamic environments for each. We check each environment to ensure that a finite policy exists prior to running the planner.

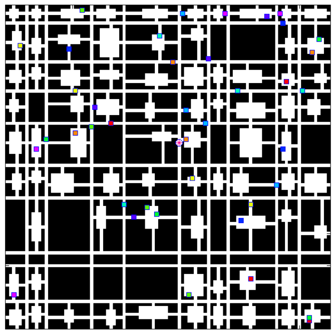


Fig. 5. A sample random 100 x 100 cell map with 10 humans.

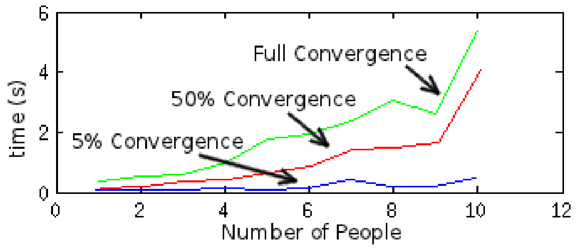


Fig. 6. Mean planning times for maps like Fig. 5. 4 trajectories per person.

People	Size	People	Size
1	$2.0 \cdot 10^6$	6	$2.2 \cdot 10^{10}$
2	$1.5 \cdot 10^7$	7	$1.3 \cdot 10^{11}$
3	$1.0 \cdot 10^8$	8	$7.0 \cdot 10^{11}$
4	$6.3 \cdot 10^8$	9	$3.9 \cdot 10^{12}$
5	$3.8 \cdot 10^9$	10	$2.1 \cdot 10^{13}$

TABLE I
SIZE OF THE BELIEF STATE SPACE FOR EACH NUMBER OF PEOPLE IN OUR EXPERIMENTS.

The belief state space for these problems is huge. For the 10 person problem, there are 100×100 cells, 11 settings of f, at least 20 timesteps (usually many more), and 5^{10} settings of H (4 trajectories plus one for $h = u$), which multiplies to a total of $21,484,375,000,000$ possible belief states. Table I shows the size for each number of people considered. This is clearly too large to represent as a generic POMDP and solve using an exact or an approximate POMDP solver, yet we can find full policies on these problems in an average of about 5 seconds as shown in the top green line in Fig. 6.

We have seen that this algorithm tends to very quickly converge to a good approximation after only a few iterations. There are an average of only 15 policy changes for an average of 41 iterations with policies that are around 200 actions long. A policy change is counted each time a previously defined entry is updated. The majority of these changes are the algorithm trying out different focus points. Therefore, if a robot started following the partial policy after a few iterations, almost all of its actions would be optimal up until the focus point. The partial policy may be incomplete because it may not yet define actions for every outcome of a stochastic transition. In these cases, it is useful to talk about the *probability of success* of the policy. This is defined as the probability that the policy will contain a complete path to the goal. Since each branch has some probability of being reached, we compute the probability of success as the probability given each unknown variable that the policy has a defined branch for that setting (this is computed efficiently by iterating over the policy only once). In Fig. 6, we show results that terminate when the probability of success reaches the stated convergence level. By planning to 50% probability of success (red line) we can save time and start executing a policy. The 5% probability of success curve shows how quickly we can get a partial solution. This is the minimum time at which the robot could start executing the policy. As it executes, planning can continue and a new policy can be followed once it is computed and has a higher probability of success than the current policy.

VII. Conclusion and Future Work

We have presented a generalization to the PPCP algorithm that allowed it to be used in cases with approximate rather than clear preferences. We explained the algorithm and a bound on the sub-optimality of the policies it produces.

We also applied this algorithm to the domain of navigation in dynamic environments with uncertain human intentions. The main contribution to this problem is the fact that we can reason about uncertainty in the trajectories of dynamic obstacles and automatically take focus actions (when available) to disambiguate between the possible future trajectory hypotheses as necessary. Our algorithm allows us to solve this planning under uncertainty problem efficiently.

To better solve this problem, one of the most important steps for the future is integrating a sophisticated multi-hypothesis dynamic obstacle tracker. We are also investigating other ways to speed up the planner by removing redundant and irrelevant states.

Another limitation of this work is that it does not model interaction between the people and the robot. When people move through an environment, they are also trying to avoid collisions, and ideally this would be included in the model. Previous work has addressed this issue, but had a much more limited model of uncertainty and is not capable of dealing with indoor environments [17].

We noticed that there appears to be a direct trade-off between the quality of the preferences (and thus the bound on optimality of the result) and the planning time. Having preferences that are "more wrong" tends to mean we ignore more people (and thus ignore more branches), and thus converge more quickly. This suggests that it may be desirable to purposely choose worse preferences if faster planning time is required.

Finally, the suboptimality bound presented in this paper is not believed to be tight, so finding a tighter bound, both on the preferences in this domain, and the PPCP algorithm overall, are promising directions of future work.

VIII. Acknowledgments

Bradford Neuman is partially supported by a Graduate Research Fellowship from the National Science Foundation.

This research was sponsored by the Army Research Laboratory Cooperative Agreement Number W911NF-10-2-0016.

We would like to thank Michael Phillips, Jon Butzke, Haojie Zhang, and Brian MacAllister for their helpful contributions and discussions.

References

[1] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in Proc. 2008 Intelligent Vehicles Symp. Eindhofen: IEEE, June 2008, pp. 1149–1154.

[2] H. Gong, J. Sim, M. Likhachev, and J. Shi, "Multi-hypothesis motion planning for visual object tracking," in Int. Conf. on Computer Vision (ICCV), 2011.

[3] B. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. D. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in Proc. 2009 Int. Conf. on Intelligent Robots and Systems, October 2009.

[4] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes," in IEEE/RSJ 2008 Int. Conf. on Intelligent Robots and Systems, Nice, France, 2008.

[5] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," Int. Journal of Robotics Research, vol. 24, no. 1, 2005.

[6] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in Proc. 6th Int. Conf. on Artificial Intelligence Planning and Scheduling, S. Chien, S. Kambhampati, and C. Knoblock, Eds. Breckenridge, CO: AAAI Press, 2000, pp. 52–61.

[7] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processses," Mathematics of Operations Research, vol. 12, no. 3, pp. 441–450, 1987.

[8] C. Baral, V. Kreinovich, and R. Trejo, "Computational complexity of planning and approximate planning in the presence of incompleteness," Artificial Intelligence, vol. 122, no. 1-2, pp. 241–267, 2000.

[9] M. Likhachev and A. Stentz, "Probabilistic planning with clear preferences on missing information," Artificial Intelligence, vol. 173, no. 56, pp. 696 – 721, 2009.

[10] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in Proc. 2011 Intl. Conf. on Robotics and Automation. IEEE, 2011, pp. 5628–5635.

[11] B. Xu, D. J. Stilwell, and A. Kurdila, "A receding horizon controller for motion planning in the presence of moving obstacles," in IEEE Int. Conf. on Robotics and Automation. IEEE, 2010, pp. 974–980.

[12] D. Silver, "Collaborative pathfinding," In Proc. of AIIDE, 2005.

[13] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in Proc. 2007 Int. Conf. on Robotics and Automation, 2007, pp. 1610–1616.

[14] E. Krotkov and R. Bajcsy, "Active vision for reliable ranging: Cooperating focus, stereo, and vergence," Int. Journal of Computer Vision, vol. 11, no. 2, pp. 187–203, 1993.

[15] J. Velez, G. Hemann, A. S. Huang, I. Posner, and N. Roy, "Planning to perceive: Exploiting mobility for robust object detection," in Int. Conf. on Automated Planning and Scheduling (ICAPS), Freiburg, Germany, Jun. 2011.

[16] M. Likhachev and A. Stentz, "PPCP: The proofs," University of Pennsylvania, Philadelphia, PA," Tech. Rep., 2008.

[17] B. Neuman and A. Stentz, "Anytime policy planning in large dynamic environments with interactive uncertainty," in Proc. 2012 Int. Conf. on Intelligent Robots and Systems. IEEE, 2012, pp. 2670–2677.