Planning-based Prediction for Pedestrians

Brian D. Ziebart[1] Nathan Ratliff[1] Garratt Gallagher[1] Christoph Mertz[1] Kevin Peterson[1]
J. Andrew Bagnell[1] Martial Hebert[1] Anind K. Dey[1] Siddhartha Srinivasa[2]

[1]*School of Computer Science*
Carnegie Mellon University
{*bziebart, ndr, ggallagh, mertz, kp, dbagnell, hebert, anind*}*@cs.cmu.edu*

[2]*Intel Research*
Pittsburgh, PA
siddhartha.srinivasa@intel.com

***Abstract*— We present a novel approach for determining robot movements that efficiently accomplish the robot's tasks while not hindering the movements of people within the environment. Our approach models the goal-directed trajectories of pedestrians using maximum entropy inverse optimal control. The advantage of this modeling approach is the generality of its learned cost function to changes in the environment and to entirely different environments. We employ the predictions of this model of pedestrian trajectories in a novel incremental planner and quantitatively show the improvement in hindrance-sensitive robot trajectory planning provided by our approach.**

I. Introduction

Determining appropriate robotic actions in environments with moving people is a well-studied [15], [2], [5], but often difficult task due to the uncertainty of each person's future behavior. Robots should certainly never collide with people [11], but avoiding collisions alone is often unsatisfactory because the disruption of almost colliding can be burdensome to people and sub-optimal for robots. Instead, robots should predict the future locations of people and plan routes that will avoid such hindrances (*i.e.*, situations where the person's natural behavior is disrupted due to a robot's proximity) while still efficiently achieving the robot's objectives. For example, given the origins and target destinations of the robot and person in Figure 1, the robot's hindrance-minimizing trajectory would take the longer way around the center obstacle (a table), leaving a clear path for the pedestrian.

One common approach for predicting trajectories is to project the prediction step of a tracking filter [9], [13], [10] forward over time. For example, a Kalman filter's [7] future positions are predicted according to a Gaussian distribution with growing uncertainty and, unfortunately, often high probability for physically impossible locations (*e.g.*, behind walls, within obstacles). Particle filters [16] can incorporate more sophisticated constraints and non-Gaussian distributions, but degrade into random walks of feasible motion over large time horizons rather than purposeful, goal-based motion. Closer to our research are approaches that directly model the policy [6]. These approaches assume that previously observed trajectories capture all purposeful behavior, and the only uncertainty involves determining to which previously observed class of trajectories the current behavior belongs. Models based on mixtures of trajectories and conditioned action distribution modeling (using hidden Markov models) have been employed [17]. This approach often suffers from over-fitting to the particular training trajectories and context of those trajectories. When changes to the environment occur (*e.g.*, rearrangement of the furniture), the model will confidently predict incorrect trajectories *through* obstacles.

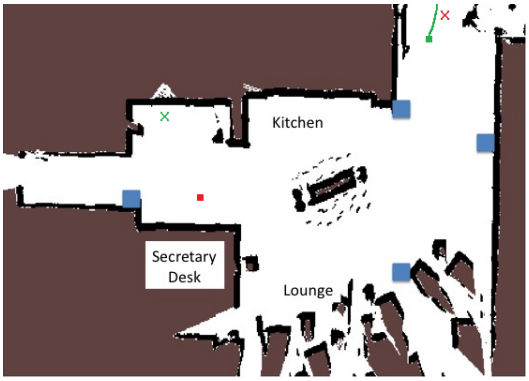

Fig. 1. A hindrance-sensitive robot path planning problem in our experimental environment containing a person (green square) in the upper right with a previous trajectory (green line) and intended destination (green X) near a doorway, and a robot (red square) near the secretary desk with its intended destination (red X) near the person's starting location. Hindrances are likely if the person and robot both take the distance-minimizing path to their intended destinations. Laser scanners are denoted with blue boxes.

Fig. 2. Images of the kitchen area (left), secretary desk area (center), and lounge area (right) of our experimental environment.

We assume that people behave like planners – efficiently moving to reach destinations. In traditional planning, given a *cost function* mapping environment features to costs, the optimal trajectory is easily obtained for any endpoints in any environment described using those features. Our approach learns the cost function that best explains previously observed trajectories. Unfortunately, traditional planning is *prescriptive* rather than *predictive* – the sub-optimality typically present in observed data is inexplicable to a planner. We employ the principle of maximum entropy to address the lack of decision uncertainty using a technique we previously developed called maximum entropy inverse optimal control (or inverse reinforcement learning) [18]. This approach yields a *soft-maximum* version of Markov decision processes (MDP) that accounts for decision uncertainty. As we shall show, this soft-max MDP model supports efficient algorithms for learning the cost function that best explains previous behavior, and for predicting a person's future positions.

Importantly, the featured-based cost function that we employ enables generalization. Specifically, the cost function is a linear combination of a given set of features computed from the environment (*e.g.*, obstacles and filters applied to obstacles). Once trained, the cost function applies to any configuration of these features. Therefore if obstacles in the environment move, the environment otherwise changes, or we consider an entirely different environment, our model generalizes to this new setting. We consider this improved generalization to be a major benefit of our approach over previous techniques.

Predictions of pedestrian trajectories can be naturally employed by a planner with time-dependent costs so that potential hindrances are penalized. Unfortunately, the increased dimensionality of the planning problem can be prohibitive. Instead, we present a simple, incremental "constraint generation" planning approach that enables real-time performance. This approach initially employs a cost map that ignores the predictions of people's future locations. It then iteratively plans the robot's trajectory in the cost map, simulates the person's trajectory, and adds "cost" to the cost map based on the probability of hindrance at each location. The time-independent cost function that this procedure produces accounts for the time-varying predictions, and ultimately yields a high quality, hindrance-free robot trajectory, while requiring much less computation than a time-based planner.

We evaluate the quality of our combined prediction and planning system on the trajectories of people in a lab environment using the opposing objectives of maximizing the robot's efficiency in reaching its intended destination and minimizing robot-person hindrances. An inherent trade-off between these two criteria exists in planning appropriate behavior. We show that for any chosen trade-off, our prediction model is better for making decisions than an alternative approach.

II. Modeling Purposeful Motion

Accurate predictions of the future positions of people enable a robot to plan appropriate actions that avoid hindering those people. We represent the sequence of actions (diagonal and adjacent movements) that lead to a person's future position using a deterministic Markov decision process (MDP) over a grid representing the environment. Unfortunately, people do not move in a perfectly predictable manner, and instead our robot must reason probabilistically about their future locations. By maximizing the entropy of the distribution of trajectories, $H(P_\zeta) = -\sum_\zeta P(\zeta)\log P(\zeta)$ subject to the constraint of matching the reward of the person's behavior in expectation [1], we obtain a distribution over trajectories [18].

In this section, we present a new interpretation of the maximum entropy distribution over trajectories and algorithms for obtaining it. This is framed as a softened version of the value iteration algorithm, which is commonly employed to find optimal policies for MDPs. We first review the Bellman equations and value iteration. We next relax these equations, obtaining a distribution over actions and trajectories. We then employ Bayes' Rule using this distribution to reason about unknown intended destinations. Next, we compute expected visitation counts, $D_{x,y}$, across the environment, and finally obtain time-based visitation counts, $D_{x,y,t}$, that can be used for hindrance-sensitive planning purposes.

A. Relaxing Maximum Value MDPs for Prediction

Consider a Markov decision process with deterministic action outcomes for modeling path planning. This class of MDPs consist of a state set, S, an action set, A, an action transition function, $T : S \times A \rightarrow S$, and a reward function, $R : S \times A \rightarrow \mathbb{R}$. A trajectory, ζ, is a sequence of states (grid cells) and actions (diagonal and adjacent moves), $\{s_0, a_0\}$ that satisfies the transition function (*i.e.*, $\forall_{s_{t+1},s_t,a_t\in\zeta}\, T(s_t, a_t) = s_{t+1}$). Goal-based planning is modeled in the MDP by assigning costs (*i.e.*, negative rewards) for every action except for a self-transitioning action in the *absorbing* goal state that has a cost of 0. MDPs are *solved* by finding the state and action values (*i.e.*, future reward of that action or state), $V^*(s)$ and $Q^*(s, a)$, for the maximum future reward policy, $\pi^* : S \rightarrow A$. The Bellman equations define the optimal quantities:

$$Q^*(s, a) = R(s, a) + V^*(T(s, a)) \quad (1)$$

$$V^*(s) = \max_a Q^*(s, a). \quad (2)$$

The value iteration algorithm produces these optimal values by alternately applying Equations 1 and 2 as update rules until the values converge. The optimal policy is then: $\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$. While useful for prescribing a set of actions to take, this policy is not usually predictive because observed trajectories are often not consistently optimal.

We employ a "softened" version of MDPs derived using the principle of maximum entropy that incorporates trajectory uncertainty into our model. In this setting, trajectories are probabilistically distributed according to their values rather than having a single optimal trajectory for the solution. We accomplish this by replacing the *maximum* of the Bellman equations with a *soft-maximum* function, $\operatorname{softmax}_x f(x) = \log \sum_x e^{f(x)}$.

$$Q^{\approx}(s, a) = R(s, a) + V^{\approx}(T(s, a)) \quad (3)$$

$$V^{\approx}(s) = \operatorname*{softmax}_a Q^{\approx}(s, a) \quad (4)$$

In this case, the solution policy is probabilistic with distribution: $\pi(a|s) = e^{Q^{\approx}(s,a)-V^{\approx}(s)}$. The probability of a trajectory, ζ, can be shown [18] to be distributed according to $P(\zeta) \propto e^{\sum_{(s,a)\in\zeta} R(s,a)}$. Trajectories with a very high reward (low cost) are exponentially more preferable to low reward (high cost) trajectories, and trajectories with equal reward are equally probable. The magnitude of the rewards, $|R(s, a)|$, is meaningful in this softmax setting and corresponds to the certainty about trajectories. As $|R(s, a)| \rightarrow \infty$, $\operatorname{softmax}_a Q^{\approx}(s, a) = \max_a Q^*(s, a)$, and the distribution converges to only optimal trajectories. An analagous $O(L|S||A|)$ time value-iteration procedure is employed (with appropriately chosen length L) to solve this softmax policy distribution – terminating when the value function has reached an acceptable level of convergence. Note that this softmax value distribution over trajectories is very different than the softmax action selection distribution that has been employed for reinforcement learning: $P_\tau(a|s) \propto e^{Q^*(s,a)/\tau}$ [14], [12].

B. Learning Unknown Cost Functions

For prescriptive MDP applications, the reward values for actions in the MDP are often engineered to produce appropriate behavior, however for our prediction purposes,

we would like to find the reward values that best predict a set of observed trajectories, $\{\tilde{\zeta}_i\}$. We assume the availability of a vector of *feature values*, $\mathbf{f}_{s,a}$, characterizing each possible action. For our application, these features are obstacle locations and functions of obstacle locations (*e.g.*, blurring and filtering of obstacles). We assume that the reward is linear in these features, $R(s,a) = \theta^\top \mathbf{f}_{s,a}$, with unknown weight parameters, θ. We denote the first state, s_0, of trajectory $\tilde{\zeta}_i$ as $s_0(\tilde{\zeta}_i)$. The learning problem is then the maximization of the observed trajectory's probability, $P(\zeta|\theta)$[1], or equivalently:

$$\theta^* = \operatorname*{argmax}_\theta \sum_i \left(\left(\sum_{(s,a)\in\tilde{\zeta}_i} \theta^\top \mathbf{f}_{s,a} \right) - V^{\approx}(s_0(\tilde{\zeta}_i)) \right). \tag{5}$$

The gradient of the function in Equation 5 has an intuitive interpretation as the difference between the feature counts of observed trajectories and expected feature counts according to the model: $\sum_{i,(s,a)\in\tilde{\zeta}_i} \mathbf{f}_{s,a} - E_{P_\theta(\zeta)}[\mathbf{f}_{s,a}]$. We employ gradient-based optimization on this convex function to obtain θ^* [18]. We refer the reader to that work for a more detailed explanation of the optimization procedure.

C. Destination Prior Distribution

Though our model is conditioned on a known destination location, that destination location is not known at prediction time. Our predictive model must reason about all possible destinations to predict the future trajectory of a person. We address this problem in a Bayesian way by first obtaining a prior distribution over destinations using previously observed trajectories and the features of the environment.

In this work, we base our prior distribution on the goals of previously observed trajectories (g). We *smooth* this probability to nearby cells using the Manhattan distance ($\text{dist}(a,b)$) and also add probability (P_0) for previously unvisited locations to avoid overfitting, yielding: $P(\text{dest } x) \propto P_0 + \sum_{\text{goals } g} e^{-\text{dist}(x,g)}$. When little or no previous data is available for a particular environment, a feature-based model of destinations with features expressing door, chair, and appliance locations could be employed.

D. Efficient Future Trajectory Prediction

In the prediction setting, the robot knows the person's partial trajectory from state A to current state B, $\zeta_{A\to B}$. and must infer the future trajectory of the person, $P(\zeta_{B\to C})$, to an unknown destination state, C, given all available information. First we infer the posterior distribution of destinations given the partial trajectory, $P(\text{dest } C|\zeta_{A\to B})$, using Bayes' Rule. For notational simplicity, we denote the softmax value function of state X to destination state Y as $V^{\approx}(X\to Y)$ and the reward of a policy as $R(\zeta) = \sum_{(s,a)\in\zeta} R(s,a)$. The posterior distribution is then:

$$\begin{aligned} P(\text{dest C}|\zeta_{A\to B}) &= \frac{P(\zeta_{A\to B}|\text{dest C})P(\text{dest C})}{P(\zeta_{A\to B})} \\ &= \frac{\frac{e^{R(\zeta_{A\to B})+V^{\approx}(B\to C)}}{e^{V^{\approx}(A\to C)}} P(\text{dest } C)}{\sum_D \frac{e^{R(\zeta_{A\to B})+V^{\approx}(B\to D)}}{e^{V^{\approx}(A\to D)}} P(\text{dest } D)}. \end{aligned} \tag{6}$$

[1]We assume the final state of the trajectory is the goal destination and our probability distribution is conditioned on this goal destination.

The value functions, $V^{\approx}(A\to D)$ and $V^{\approx}(B\to D)$, for each state D are required to compute this posterior (Equation 6). The naïve approach is to execute $O(|D|)$ runs of softmax value iteration – one for each possible goal, D. Fortunately there is a much more efficient algorithm. In the hard maximum case, this problem is solved efficiently by modifying the Bellman equations to operate backwards, so that instead of $V(S)$ representing the future value of state S, it is the maximum value obtained by a trajectory reaching state S. Initializing $\forall_{s\neq A} V(s) = -\infty$ and $V(A) = 0$, the following equations define $V(A\to D)$ for all D.

$$Q(s,a) = R(s,a) + V(s) \tag{7}$$

$$V(s) = \max_{(s',a):T(s',a)=s} Q(s',a) \tag{8}$$

For the soft-max reward case, the $\max$ is replaced with softmax (Equation 8) and the value functions, $V^{\approx}(A\to D)$, are obtained with a value-iteration algorithm. Thus, with two applications of value-iteration to produce $V^{\approx}(A\to D)$ and $V^{\approx}(B\to D)$, and $O(|D|)$ time to process the results, the posterior distribution over destinations is obtained.

We now use the destination posterior to compute the conditional probability of any continuation path, $\zeta_{B\to C}$:

$$\begin{aligned} &P(\zeta_{B\to C}|\zeta_{A\to B}) \\ &\quad= \sum_D P(\zeta_{B\to C}|\zeta_{A\to B}, \text{dest } D) P(\text{dest } D|\zeta_{A\to B}) \\ &\quad= P(\zeta_{B\to C}|\text{dest } C) P(\text{dest } C|\zeta_{A\to B}) \\ &\quad= e^{R(\zeta_{B\to C}) - V^{\approx}(B\to C)} P(\text{dest } C|\zeta_{A\to B}). \end{aligned} \tag{9}$$

This can be readily computed using the previously computed posterior destination distribution and $V^{\approx}(B\to C)$ is computed as part of generating that posterior distribution.

The expected occupancies of different states, D_x, are obtained by marginalizing over all paths containing D_x. For this class of paths, denoted $\Xi_{B\to x\to C}$, the path can be divided[2] into a path from B to x and a path from x to C.

$$\begin{aligned} D_s &= \sum_C \sum_{\zeta\in\Xi_{B\to x\to C}} P(\zeta_{B\to C}|\text{dest } C) P(\text{dest } C|\zeta_{A\to B}) \\ &= \sum_C P(\text{dest } C|\zeta_{A\to B}) \sum_{\substack{\zeta_1\in\Xi_{B\to x},\\ \zeta_2\in\Xi_{x\to C}}} e^{R(\zeta_1)+R(\zeta_2)-V^{\approx}(B\to C)} \\ &= \left(\sum_{\zeta_1\in\Xi_{A\to x}} e^{R(\zeta_1)} \right) \\ &\quad \left(\sum_C \sum_{\zeta_2\in\Xi_{x\to C}} e^{R(\zeta_2)+\log P(\text{dest } C|\zeta_{A\to B})-V^{\approx}(B\to C)} \right) \end{aligned} \tag{10}$$

The first summation of Equation 10 equates to $e^{V^{\approx}(A\to x)}$, which is easily obtained from previously computed value functions. We compute the second double summation by adding a final state reward of $(\log P(\text{dest } C|\xi_{A\to B}) - V^{\approx}(B\to C))$ and performing soft value iteration with those modified rewards. Thus with one additional application of the soft value iteration algorithm and combining the results (constant time with respect to the number of goals), we obtain state expected visitation counts.

[2]The solution also holds for paths with multiple occurances of a state.

Algorithm 1 Incorporating predictive pedestrian models via predictive planning

procedure PREDICTIVEPLANNING($\sigma > 0$, $\alpha > 0$, $\{D_{s,t}\}$, D_{thresh})
  Initialize cost map to prior navigational costs $c_0(s)$.
  for $t = 0, \ldots, T$ **do**
    Plan under the current cost map.
    Simulate the plan forward to find points of probable interference with the pedestrian $\{(s_i)\}_{i=1}^{K_t}$ where $D_{s,t} > D_{\text{thresh}}$.
    If $K = 0$ then break.
    Add cost to those points
      $c_{t+1}(s) = c_t(s) + \alpha \sum_{i=1}^{K_t} e^{-\frac{1}{2\sigma^2}\|s - s_i\|^2}$.
  end for
  return The plan through the final cost map.
end procedure

E. Temporal Predictions

To plan appropriately requires predictions of where people will be at different points in time. More formally, we need predictions of expected future occupancy of each location during the time windows surrounding fixed intervals: $\tau, 2\tau, ..., T\tau$. We denote these quantities as $D_{s,t}$. In theory, time can be added to the state space of a Markov decision process and explicitly modeled. In practice, however, this expansion of the state space significantly increases the time complexity of inference, making real-time applications based on the time-based model impractical. We instead consider an alternative approach that is much more tractable.

We assume that a person's movement will "consume" some cost over a time window t according to the normal distribution $N(tC_0, \sigma_0^2 + t\sigma_1^2)$, where C_0, σ_0^2, and σ_1^2 are learned parameters. Certainly $\sum_t D_{s,t} = D_s$, so we simply divide the expected visitation counts among the time intervals according to this probability distribution. We use the cost of the optimal path to each state, $Q^*(s)$, to estimate the cost incurred in reaching it. The resulting time-dependent occupancy counts are then:

$$D_{s,it} \propto D_s e^{\frac{-(C_0 t - Q^*(s))^2}{2(\sigma_0^2 + t\sigma_1^2)}}. \tag{11}$$

These values are computed using a single execution of Dijkstra's algorithm [3] in $O(|S| \log |S|)$ time to compute $Q^*(.)$ and then $O(|S|T)$ time for additional calculation.

III. PLANNING WITH PEDESTRIAN PREDICTIONS

Ideally, to account for predictive models of pedestrian behavior, we should increase the dimensionality of the planning problem by augmenting the state of the planner to account for time-varying costs. Unfortunately, the computational complexity of combinatorial planning is exponential in the dimension of the planning space, and the added computational burden of this solution will be prohibitive for many real-time applications.

We therefore propose a novel technique for integrating our time-varying predictions into the robot's planner. Algorithm 1 details this procedure; it essentially iteratively shapes a time-independent navigational cost function to remove known points of hindrance. At each iteration, we run the time-independent planner under the current cost map and simulate forward the resulting plan in order to predict points at which the robot will likely interfere with the pedestrian. By then adding cost to those regions of the map we can ensure that subsequent plans will not interfere at those locations. We can further improve the computational gain of this technique by using efficient replanners such as D* and its variants [4] in the inner loop. While this technique, as it reasons only about stationary costs, cannot guarantee the optimal plan given the time-varying costs, we demonstrate that it produces good robot behavior in practice that efficiently accounts for the predicted motion of the pedestrian.

By re-running this iterative replanner every 0.25 seconds using updated predictions of pedestrian motion, we can achieve intelligent adaptive robot behavior that anticipates where a pedestrian is heading and maneuvers well in advance to implement efficient avoidance. The accompanying movie demonstrates the behavior that emerges from our predictive planner in select situations. In practice, we use the final cost-to-go values of the iteratively constructed cost map to implement a policy that chooses a good action from a predefined collection of actions. When a plan with sufficiently low probability of pedestrian hindrance cannot be found, the robot's speed is varied. Additionally, when the robot is too close to a pedestrian, all actions that take the robot within a small radius of the human are removed to avoid potential collisions. Section IV-F presents quantitative experiments demonstrating the properties of this policy.

IV. EXPERIMENTAL EVALUATION

We now present experiments demonstrating the capabilities of our prediction model and its usefulness for planning hindrance-sensitive robot trajectories.

A. Data Collection

We collected over one month's worth of data in a lab environment. The environment has three major areas (Figure 1): a kitchen area with a sink, refrigerator, microwave, and coffee maker; a secretary desk; and a lounge area. We installed four laser range finders in fixed locations around the lab, as shown in Figure 1, and ran a pedestrian tracking algorithm [8]. Trajectories were segmented based on significant stopping time in any location.

Fig. 3. Collected trajectory dataset.

From the collected data, we use a subset of 166 trajectories through our experimental environment to evaluate our approach. This dataset is shown in Figure 3 after post-processing and being fit to a 490 by 321 cell grid (each cell represented as a single pixel). We employ 50% of this data

as a training set for estimating the parameters of our model and use the remainder for evaluative purposes.

B. Learning Feature-Based Cost Functions

We learn a 6-parameter cost function over simple features of the environment, which we argue are easily transferable to other environments. The first feature is a constant feature for every grid cell in the environment. The remaining functions are an indicator function for whether an obstacle exists in a particular grid cell, and four "blurs" of obstacle occupancies, which are shown in Figure 4.

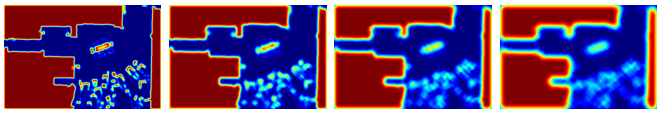

Fig. 4. Four obstacle-blur features for our cost function. Feature values range from low weight (dark blue) to high weight (dark red).

We then learn the weights for these features that best explain the demonstrated data. The resulting cost function for the environment is shown in Figure 5. Obstacles in the cost function have very high cost, and free space has a low cost that increases near obstacles.

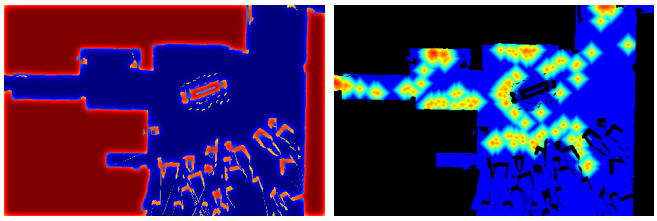

Fig. 5. Left: The learned cost function in the environment. Right: The prior distribution over destinations learned from the training set.

The prior distribution over destinations is obtained from the set of endpoints in the training set, and the temporal Gaussian parameters are also learned using the training set.

C. Stochastic Modeling Experiment

We first consider two examples from our dataset (Figure 6) that demonstrate the need for uncertainty-based modeling.

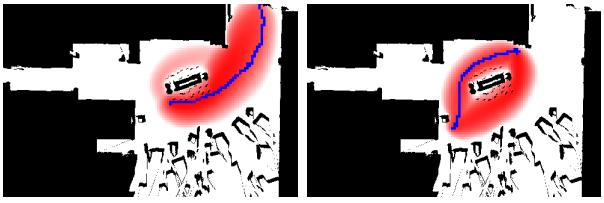

Fig. 6. Two trajectory examples (blue) and log occupancy predictions (red).

Both trajectories travel around the table in the center of the environment. However, in the first example (left), the person takes the lower pathway around the table, and in the second example (right), the person takes the upper pathway despite that the lower pathway around the table has a lower cost in the learned cost function. In both cases, the path taken is not the shortest path through the open space that one would obtain using an optimal planner. Our uncertainty-based planning model handles these two examples appropriately, while a planner would choose one pathway or the other around the table and, even after smoothing the resulting path into a probability distribution, tend to get a large fraction of its predictions wrong when the person takes the "other" approximately equally desirable pathway.

D. Dynamic Feature Adaptation Experiment

In many environments, the relevant features that influence movement change frequently – furniture is moved in indoor environments, the locations of parked vehicles are dynamic in urban environments, and weather conditions influence natural environments with muddy, icy, or dry conditions. We demonstrate qualitatively that our model of motion is robust to these feature changes.

The left frames of Figure 7 show the environment and the path prediction of a person moving around the table at two different points in time. At the second point of time (bottom left), the probability of the trajectory leading to the kitchen area or the left hallway is extremely small. In the right frames of Figure 7, an obstacle has been introduced that blocks the direct pathway through the kitchen area. In this case, the trajectory around the table (bottom right) still has a very high probability of leading to either the kitchen area or the left hallway. As this example shows, our approach is robust to changes in the environment such as this one.

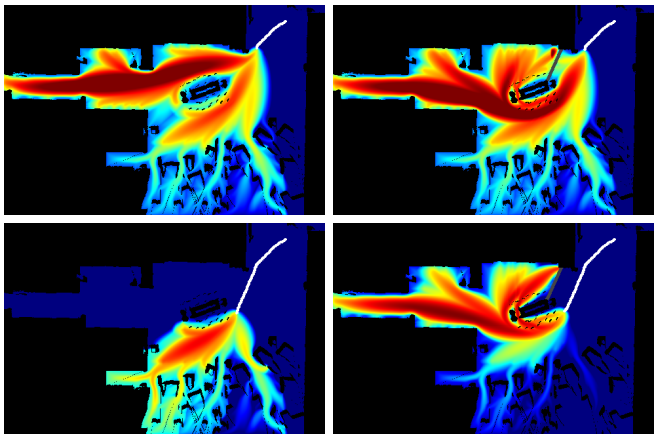

Fig. 7. Our experimental environment with (right column) and without (left column) an added obstacle (gray) between the kitchen and center table. Predictions of future visitation expectations given a person's trajectory (white line) in both settings for two different trajectories. Frequencies range from red (high log expectation) to dark blue (low log expectation).

E. Comparative Evaluation

We now compare our model's ability to predict the future path of a person with a previous approach for modeling goal-directed trajectories – the variable-length Markov model (VLMM) [6]. The VLMM estimates the probability of a person's next cell transition conditioned on the person's history of cells visited in the past. It is variable length because it employs a long history when relevant training data is abundant, and a short history otherwise.

The results of our experimental evaluation are shown in Figure 8. We first note that for the training set (denoted *train*), that the trajectory log probability of the VLMM is significantly better than the plan-based model. However, for

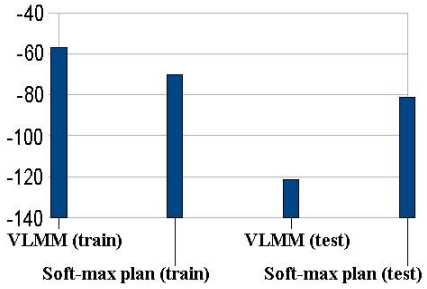

Fig. 8. Log probability of datasets under the VLMM and our approach.

the test set, which is the metric we actually care about, the performance of the VLMM degrades significantly, while the degradation in the plan-based model is much less extreme. We conclude from this experiment that the VLMM (and similar directed graphical model approaches) are generally much more difficult to train to generalize well because their number of parameters is significantly larger than the number of parameters of the cost function employed in our approach.

F. Integrated Planning Evaluation

We now simulate robot-human hindrance problems to demonstrate the benefit of our trajectory forecasting approach. We generate 200 hindrance-sensitive planning problems (corresponding to 22 different person trajectories in Figure 3) by selecting problems where naïve planning (disregarding the pedestrian) causes hindrances. We ignore the causal influence of the robot's action on the person's trajectory, and measure the trade-off between robot efficiency and interference with the person's trajectory. Specifically, the *average hindrance count* measures the average number of times the policy removed actions due to proximity to a human, and the *average execution time* measures the number of time steps needed to reach the goal. The trade-off is controlled by varying the degree of the visitation frequency threshold used in Algorithm 1. The robot trajectory planner is provided with person location forecasts at 4Hz.

The trade-off curves of planning using our plan-based forecasts and using a straight-forward particle-based forecasting model on this set of problems are shown in Figure 9. For both predictive models, while the execution time varies over a small range, the average hindrance count decreases by a factor of two. Additionally, as the figure shows, the plan-based forecasting is superior to the particle-based approach for almost any level of the trade-off.

V. Conclusions and Future Work

We have presented a novel approach for predicting future pedestrian trajectories using a soft-max version of goal-based planning. As we have shown, the feature-based cost function learned using this approach allows accurate generalization to changes in the environment. We additionally showed the usefulness of this approach for planning hindrance-sensitive routes using a novel incremental path planner. In future work, we plan to explicitly model interactions between people so that we can better predict movements in crowded environments. Additionally, we have assumed a fully observed world that many robotics applications lack. Effeciently extending our approach to the setting where human behavior is only partially observable remains as an important extension.

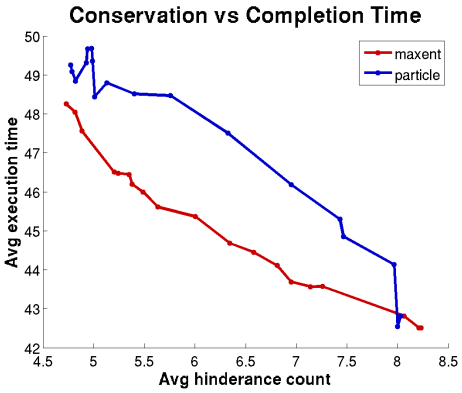

Fig. 9. The tradeoff in efficiency versus pedestrian hindrance for varying degrees of hindrance penalization for planning under both planning-based predictions and particle-based predictions.

VI. Acknowledgements

This material is based upon work partially supported by the National Science Foundation under Grant No. EEC-0540865, the NSF Graduate Research Fellowship program, a DARPA Learning for Locomotion Contract, and by Intel Research Pittsburgh.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, pages 1–8, 2004.
[2] M. Bennewitz, W. Burgard, and S. Thrun. Learning motion patterns of persons for mobile service robots. In *Proc. ICRA*, pages 3601–3606, 2002.
[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
[4] D. Ferguson and A. T. Stentz. Field D*: An interpolation-based path planner and replanner. In *Proc. ISRR*, pages 239–253, 2005.
[5] A. F. Foka and P. E. Trahanias. Predictive autonomous robot navigation. In *Proc. IROS*, pages 490–495, 2002.
[6] A. Galata, N. Johnson, and D. Hogg. Learning variable-length Markov models of behavior. *Computer Vision and Image Understanding*, 81(3):398–413, 2001.
[7] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *AMSE Journ. Basic Eng.*, pages 95–108, 1962.
[8] R. MacLachlan. Tracking moving objects from a moving vehicle using a laser scanner. Technical Report CMU-RI-TR-05-07, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2005.
[9] R. Madhavan and C. Schlenoff. Moving object prediction for off-road autonomous navigation. In *SPIE Aerosense Conference*, 2003.
[10] C. Mertz. A 2d collision warning framework based on a Monte Carlo approach. In *Proc. of Intelligent Transportation Systems*, 2004.
[11] S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *Proc. IROS*, pages 3726–3731, 2005.
[12] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI*, pages 2586–2591, 2007.
[13] C. Schlenoff, R. Madhavan, and T. Barbera. A hierarchical, multi-resolutional moving object prediction approach for autonomous on-road driving. *Proc. ICRA*, 2:1956–1961, 2004.
[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.
[15] S. Tadokoro, M. Hayashi, Y. Manabe, Y. Nakami, and T. Takamori. On motion planning of mobile robots which coexist and cooperate with human. In *Proc. IROS*, pages 518–523, 1995.
[16] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.
[17] D. A. Vasquez Govea, T. Fraichard, O. Aycard, and C. Laugier. Intentional motion on-line learning and prediction. In *Proc. Int. Conf. on Field and Service Robotics*, pages 411–425, 2005.
[18] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1439, 2008.