Multi-Robot Inverse Reinforcement Learning under Occlusion with Interactions

Kenneth Bogert
Computer Science Department, University of Georgia
kbogert@uga.edu

Prashant Doshi
Computer Science Department, University of Georgia
pdoshi@cs.uga.edu

ABSTRACT

We consider the problem of learning the behavior of multiple mobile robots executing fixed trajectories in a common space and possibly interacting with each other in their execution. The mobile robots are observed by a subject robot from a vantage point from which it can observe a portion of their trajectories only. This problem exhibits wide-ranging applications and the specific application we consider here is that of the subject robot who desires to penetrate a *simple* perimeter patrol by two interacting robots and reach a goal location. Our approach extends single-agent inverse reinforcement learning (IRL) to a multi-robot setting and partial observability, and models the interaction between the mobile robots as equilibrium behavior. IRL provides weights over the features of the robots' reward functions, thereby allowing us to learn their preferences. Subsequently, we derive a Markov decision process based policy for each other robot. We extend a predominant IRL technique and empirically evaluate its performance in our application setting. We show that our approach in the application setting results in significant improvement in the subject's ability to predict the patroller positions at different points in time with a corresponding increase in its successful penetration rate.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems ; I.2.9 [**Robotics**]: Workcell organization and planning

General Terms

Algorithms, Performance

Keywords

inverse reinforcement, machine learning, multi-robot systems, patrolling

1. INTRODUCTION

We consider the problem of learning the behavior of multiple mobile robots executing fixed trajectories in a common space and possibly interacting with each other in their executions. A subject robot who observes the mobile robots from a vantage point from which a significant portion of their trajectories is occluded, is tasked with learning their behaviors. This problem has wide-ranging applications in robotics including forming an ad hoc team by coordinating with the other robots, executing a follow-the-leader behavior, and penetrating a *simple* perimeter patrol of two interacting robots in order to reach a goal location.

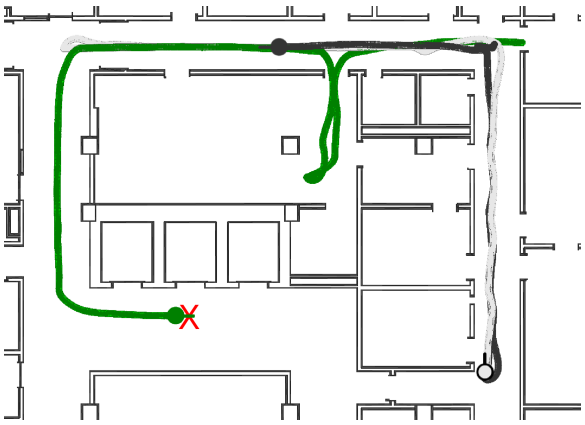

Figure 1: An example run of the subject robot in green reaching the goal state in the presence of two patrollers executing simple cyclic trajectories, shown in dark and light gray.

Inverse reinforcement learning (IRL) [11] investigates ways by which a learner may approximate the preferences of an expert by observing the experts' actions over time. Usually, the expert is assumed to be optimizing its actions using a Markov decision process (MDP), whose parameters except for the reward function are known to the learner. The reward function is commonly modeled as a linear combination of feature functions. This reduces the IRL problem to that of finding the weights in the combination, which completes the MDP, such that the optimal policy from the MDP matches the observations.

Consequently, IRL methods provide an approach to learn the behaviors of the mobile robots from observations. However, previous applications of IRL techniques in robotics such as in learning from demonstrations [4] and in other areas [1] have been in situations where the results from our physical runs reflect those from simulations as the. Our problem limits the subject robot to observing a portion of the state space that is occupied by multiple mobile robots, whose trajectories may be disturbed due to interactions.

In this paper, we present approaches that extend IRL to simultaneously learn the behavior of multiple robots in our problem context. Specifically, we modify a predominant IRL method – a maximum entropy-based approach [17] – to the context of obtaining observations over a portion of the state space only and where the behavior of the other robots is disturbed due to interactions. While we could model the joint state of the robots in a single large

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

MDP taking interactions into account in the transition function, the joint MDP becomes both prohibitively large and leads to a partially observable state space when both robots are not observable at the same time. Consequently, we model each robot separately except for when they interact. At these points of interaction, we model the multi-robots playing a game with the interaction behavior being one of possibly multiple Nash equilibria of the game.

We evaluate our approach in the application setting of *two* mobile robots executing simple cyclic trajectories for perimeter patrolling with the subject robot observing them from a point that affords partial observability of their trajectories only. The subject robot learns a set of policies that best matches the observed motion of the patrollers. It then simulates these policies forward in time, starting at the point of the recent observations while accounting for the interactions. This facilitates an informed prediction of the future positions of each patroller in space and time in order to plan a path through the space that avoids detection. We evaluate our algorithms by experimenting with both simulated and physical robots. Our results demonstrate significantly more accurate predictions of future patroller positions and a corresponding increase in the rate of successful penetrations across various degrees of observability, in comparison to using standard IRL without modeling interactions. We show an example run in Fig. 1 where the subject robot reaches the goal without being detected by the two patrollers. In some cases, the subject robot performs as well as the upper bound in which it has perfect knowledge of the patrollers' policies and how they interact.

2. BACKGROUND ON INVERSE REINFORCEMENT LEARNING

Inverse reinforcement learning [11] seeks to find the most likely policy, π_I, that an expert, I, is executing. IRL methods usually assume the presence of a single expert, that the expert has solved a MDP, and that this MDP excluding the reward function is known to the learner.

Because the space of possible reward functions is very large, it is common to express the function as a linear combination of *feature functions*. It is defined as a function, ϕ: $S \times A_I \rightarrow \mathbb{R}$, which maps a state from the set of states, S, and an action from the set of I's actions, A_I, to a real number. It is based on observable aspects of the environment, and in general, all features must be observable to both the expert and the learner. The expert's reward function is then approximated by a linear combination of $K > 0$ feature functions, $R_I(s,a) = \sum_K \theta_k \cdot \phi_k(s,a)$, where θ_k are the weights.

In order to evaluate the quality of the learned policy, many IRL algorithms make use of feature expectations. For a learned policy, π_I, the k^{th} feature expectation is, $\sum_s \mu_{\pi_I}(s) \times \phi_k(s, \pi_I(s))$. Here, $\mu_{\pi_I}(s)$ is the number of times state, s, is visited on using the policy, π_I, and may be computed using dynamic programming:

$$\mu_{\pi_I}(s) = \mu^0_{\pi_I}(s) + \gamma \sum_{s'} T(s, \pi_I(s), s') \mu_{\pi_I}(s') \qquad (1)$$

where, $\mu^0_{\pi_I}$ is initialized to 0 for all states, $0 < \gamma < 1$ is the discount factor, $T : S \times A \times S' \rightarrow [0,1]$ is the transition function.

These feature expectations are compared with those of the expert's from its observed trajectory, which are obtained as, $\sum_{s,a \in traj} \phi_k(s,a)$. Let Π_I be the space of expert's policies whose size is $|A_I|^{|S|}$. IRL seeks to learn a policy, $\pi^*_I \in \Pi_I$, which minimizes the difference between the two expectations. More observations of the expert improve the estimate of its feature expectations.

2.1 Maximum Entropy

Often, multiple policies may match the observed feature expectations equally well. In order to resolve this ambiguity, the principle of *maximum entropy* is useful. It allows maintaining a distribution over the policies constrained to match the observed feature expectations while not being committed to any policy except as required by the constraints.

We may formulate the problem as one of finding a distribution over deterministic policies, $Pr(\Pi_I)$, which has the maximum entropy while matching the feature expectations from each policy with those observed from the expert's trajectory. Mathematically, we define the problem as a nonlinear optimization:

$$\max_{\Delta} \left(- \sum_{\pi_I \in \Pi_I} Pr(\pi_I)\, log Pr(\pi_I) \right)$$

$$\text{subject to} \qquad (2)$$

$$\sum_{\pi_I \in \Pi_I} Pr(\pi_I) = 1$$

$$\sum_{\pi_I \in \Pi_I} Pr(\pi_I) \sum_{s \in S} \mu_{\pi_I}(s) \phi_k(s, \pi_I(s)) = \hat{\phi}_k \quad \forall k$$

Here, Δ is the space of all distributions, $Pr(\Pi_I)$; $\hat{\phi}_k$ is the expectation over the k^{th} feature from observations of the expert; and the visitation frequency, μ_{π_I}, is computed as in Eq. 1.

In order to solve this nonlinear program, we may apply the Lagrangian relaxation technique bringing both the constraints into the objective function and solving the dual Lagrangian. Let the Lagrangian function be $\mathcal{L}(Pr, \eta, \boldsymbol{\theta})$ where η and $\boldsymbol{\theta}$ are the Lagrange multipliers. We obtain its partial derivative w.r.t. $Pr(\pi_I)$ and $\boldsymbol{\theta}$ as:

$$\frac{\partial \mathcal{L}}{\partial Pr(\pi_I)} = \sum_s \mu_{\pi_I}(s) \sum_k \theta_k \phi_k(s, \pi_I(s)) - log Pr(\pi_I) + \eta - 1 \qquad (3)$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \sum_{\pi_I \in \Pi_I} Pr(\pi_I) \sum_k \sum_s \mu_{\pi_I}(s) \phi_k(s, \pi_I(s)) - \hat{\phi}_k \qquad (4)$$

The objective function in (2) may have its optimal solution as a saddle point in $\mathcal{L}$. However, many numerical optimization techniques such as hill climbing and gradient descent are instead designed to find local minima or maxima. Subsequently, we modify the objective function of the relaxed Lagrangian so that the optimal solution(s) reside at local or global maxima or minima. The revised objective function is $\sqrt{\frac{\partial \mathcal{L}}{\partial Pr(\pi_I)}^2 + \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}^2}$.

2.2 Approximating Maximum Entropy

Dvijotham and Todorov [9] suggests an approach for approximating Eq. 3 that avoids computing η. Notice that the term, $\mu_{\pi_I}(s) \sum_k \theta_k \times \phi_k(s, \pi_I(s))$, in Eq. 3 represents the action value of performing I's policy, π_I, from state, s, which is commonly denoted as $Q_I(s, \pi_I(s))$. We desire to learn an MDP whose optimal solution prescribes actions that matches the observed actions of the expert. Subsequently, we may approximate Eq. 3 with:

$$\frac{\partial \mathcal{L}}{\partial Pr(\pi_I)} \approx \sum_{(s,a) \in traj} Q_{\pi_I}(s,a) - V_{\pi_I}(s) \qquad (5)$$

where, $Q_{\pi_I}(s,a)$ denotes the action value of performing the observed action, a, from state, s, and following π_I thereafter; and $V_{\pi_I}(s)$ is the optimal value of (candidate) I's policy, π_I. Note that $Q_{\pi_I}(s,a) - V_{\pi_I}(s) = 0$ when $a = \pi_I(s)$. In other words, the value difference is 0 when the action from the policy under consideration

matches the observed action. Then, we have found a reward function that leads to the MDP whose solution matches observations. However, it is possible that some other action could also result in a Q-value that is the same as the optimal value. Therefore, the above substitution is an approximation. If the observed action at s fails to match the policy, the difference is negative and accumulates over the trajectory.

3. IRL FOR MULTIPLE MOBILE ROBOTS

As we mentioned previously, this paper focuses on the problem of learning the behavior of $N \geq 2$ mobile robots executing fixed trajectories in a common space and possibly interacting with each other in their execution. Consequently, the maximum entropy based method outlined in Section 2 must be generalized in multiple ways. While the mobile robots will take on the role of being the experts in IRL, a key generalization is to learn the reward functions of multiple experts who could be interacting.

A straightforward approach would be to model the other robots jointly in a single MDP whose joint state includes the local state of each robot and a variable to allow for distinguished states where the interaction occurs. The action space is the Cartesian product of the set of actions of each robot, which includes the actions each robot may perform during interactions. The transition function of this MDP gives the next joint state due to the robots performing the joint action from a current state. The reward function gives the preferences of performing each joint action from each joint state.

This approach suffers from two significant limitations, which precludes its usage: (i) The joint state space may get very large and the transition function models the effect of performing all actions from each state including the effects of performing actions related to the interaction from non-interacting states as well. This not only increases the size of the MDP but its solution must consider the effect of performing each joint action from each state as well, which significantly increases the time to solve it optimally. Note that several potential MDPs are solved during IRL, and therefore, the learning may be slowed significantly. (ii) Traditional IRL is targeted toward environments exhibiting complete observability of the expert's actions. As a result, all robots must be visible at all times to the learner in order to observe the joint action from the joint state. However, this may not be possible in the context of mobile robots executing their own trajectories parts of which may be occluded. Indeed, this complicates the learning procedure significantly, and as shown by Choi and Kim [8], the model becomes a POMDP.

3.1 IRL for Robots with Occlusion

Given the limitations above, we pursue an alternative approach of ascribing a smaller, individual MDP to each mobile robot. Behaviors at the interacting states are modeled separately and override those prescribed by the individual MDPs at the interacting states. Although we are solving more MDPs, each is significantly smaller than the joint in its state and action spaces, and as we show in our experiments solving takes much less time. Subsequently, a nonlinear program that combines the one in (2) for each mobile robot is used for learning the policies of the robots:

$$\max_{\Delta_1,\ldots,\Delta_N} \quad -\sum_{n=1}^{N} \sum_{\pi_n \in \Pi_n} Pr(\pi_n)\, log Pr(\pi_n) \tag{6}$$

subject to

$$\begin{aligned}
&\sum_{\pi_1 \in \Pi_1} Pr(\pi_1) = 1\,,\ \ldots\ , \sum_{\pi_N \in \Pi_N} Pr(\pi_N) = 1 \\
&\sum_{\pi_1 \in \Pi_1} Pr(\pi_1) \sum_{s \in S} \mu_{\pi_1}(s)\phi_k(s, \pi_1(s)) = \hat{\phi}_{k,1} \qquad \forall k \\
&\vdots \\
&\sum_{\pi_N \in \Pi_N} Pr(\pi_N) \sum_{s \in S} \mu_{\pi_N}(s)\phi_k(s, \pi_N(s)) = \hat{\phi}_{k,N} \qquad \forall k
\end{aligned} \tag{7}$$

Another complication in our problem that is not usually present in previous applications of IRL such as learning from demonstrations and others, is that significant portions of the state space of the mobile robots may be occluded from the subject robot. Let $Obs(S) \subseteq S$ be the subset of all states of each robot that are observable. Policy π_n is then obtained by limiting the optimization over the observable state space only because we are unable to compute the observed feature expectations for the occluded states. While feature expectations from the observed states could be projected to the occluded states, this approach may not account for any disturbances in the trajectory. To account for the partial observability, we revise Eq. 4 as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \sum_{\pi_n \in \Pi_n} Pr(\pi_n) \sum_k \sum_{s \in Obs(S)} \mu_{\pi_n}(s)\phi_k(s, \pi_n(s)) - \hat{\phi}_k \tag{8}$$

In addition, we sum over the partial subset of each of the other robot's trajectory that is observable, in Eq. 5. The above revisions to the equations have the effect of under-constraining the optimization problem compared to the original.

While Broyden-Flecher-Goldfarb-Shanno (BFGS) [6] – a fast gradient descent technique – is the method of choice for solving the relaxed Lagrangian for the maximum entropy method, it may not be utilized here. This is because the gradient is not defined for the unobservable states (Eq. 8) due to which BFGS may fail to learn the Lagrange multipliers for those feature functions that apply in the occluded states only. A method that does not use the gradient function such as Nelder-Mead's simplex technique [10] is more suitable when some states are occluded.

3.2 Modeling the Interaction

At certain states where the robots interact, their behavior requires coordination. For example, consider the application domain of two mobile robots, I and J, executing a simple perimeter patrol. If the common space is narrow as in a corridor, the robots must coordinate their actions when they are in close proximity to minimize the disturbance to their patrols.

	Sidestep	Turn left	Turn right	Turn around	Stop
Sidestep	**5,5**	5,1	5,1	**5,5**	**5,5**
Turn left	1,5	1,1	1,1	1,2	1,0
Turn right	1,5	1,1	1,1	1,2	1,0
Turn around	**5,5**	2,1	2,1	2,2	2,0
Stop	**5,5**	0,1	0,1	0,2	0,0

Table 1: An example game that models the interaction between two patrolling robots (row player is I, column player is J) that approach each other from opposite directions in a corridor. The Nash equilibria of the game (shown in bold) are the possible ways of resolving this conflict. Values are a function of the agents' effort and outcome of the interaction.

At interaction states such as the one in Fig. 2, the subject robot, L, models the robots as playing a game. The strategies of this game

correspond to the actions in each robot's MDP. The solution of a game is a Nash equilibrium, which is a profile of strategies with the property that no player has an incentive to deviate from its strategy given the other's strategy in the profile. A game may admit multiple profiles in equilibrium. The robots adopt a profile of strategies in equilibrium as a way to resolve any conflicts during the interaction. If the interaction is observed, robot L constructs an example game whose sole equilibrium is the observed one. On the other hand, if the interaction behavior is occluded, robot L constructs a game whose equilibria include all the possible profiles of strategies that resolve the conflict. In Table 1, we show an example game in the context of the patrolling application domain, which has five profiles in equilibria each representing a possible way of resolving the difficulty in moving when the patrolling robots are approaching each other from opposite directions.

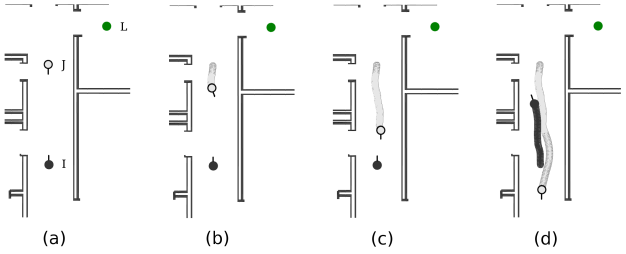

Figure 2: (a) **Patrollers I and J spot each other approaching thereby entering an interaction state.** (b) **The two robots begin executing joint actions in equilibrium.** (c) **Here, I stops while J sidesteps.** (d) **Interaction behavior is completed. The two robots continue with their trajectories as guided by their policies.**

In order to accurately learn the policy of each robot, the interaction behavior as prescribed by an equilibrium must be considered during IRL. Otherwise, because the interactions affect the observed behavior, the policy learned will likely be different from the actual policy of the observed mobile robot. *Interactions impact the state-visitation frequencies of the robots.* Let, $\sigma^e = \langle \sigma_1, \ldots, \sigma_N \rangle$ be an equilibrium with σ_n, $n = 1, \ldots, N$ denoting each robot's actions that are in equilibrium, respectively.

We may decompose Eq. 1 into a piecewise function for each other robot, $n = 1, \ldots, N$, as,

$$\mu_{\pi_n}(s) = \begin{cases} \mu^0_{\pi_n}(s) + \gamma \sum_{s'} T(s, \pi_n(s), s') \mu_{\pi_n}(s') & \text{if } s \text{ is not an interacting state} \\ \mu^0_{\pi_n}(s) + \gamma \sum_{s'} T(s, \sigma_n(s), s') \mu_{\pi_n}(s') & \text{if } s \text{ is an interacting state} \end{cases} \quad (9)$$

We may then replace μ_{π_I} in Eqs. 3 and 4 with the above equation.

Notice that the new state-visitation frequency computation requires the subject robot to ascertain whether a particular state is an interacting state. This is possible if the entire state space is observable to L. On the other hand, if a portion of the state space is occluded from L, it may not observe when and where interaction(s) between the robots occurs. Consequently, the learned policy may not be accurate and the above principled approach of Eq. 9 is not robust in the context of occlusion of the state space.

Instead, we empirically compute the state-visitation frequencies by projecting in the full environment the current policy under consideration for each robot for a large number of time steps overlaid with the equilibrium behavior, σ^e, when the robots interact. The state visitations in the projections are accumulated to obtain the approximate state-visitation frequency, denoted as $\mu^e_{\pi_n}$. Our next step is to utilize this empirical function in the constraints of the nonlinear optimization of (7):

$$\sum_{s \in \text{Obs}(S)} \mu^e_{\pi_n}(s) \phi_k(s, \pi_n(s)) = \hat{\phi}_k \quad \forall k \quad (10)$$

3.3 Multiple Equilibria

While the interaction game may admit multiple equilibria, the robots pick one to resolve any conflict during each interaction. If the interaction is not observed, how should the subject robot, L, model which equilibrium behavior was used?

Our approach is to retain each equilibrium behavior and weight its potential based on how close each behavior gets the projections to the observations. If $\mu^e_{\pi_n}$ is the empirical state-visitation frequency when equilibrium, σ^e, is used at the interaction and π_n elsewhere, and $\hat{\mu}_{\pi_n}$ is the state-visitation frequencies from the observed trajectories, then L weights the potential of this interaction behavior as an inverse function of the difference, $\omega^e = e^{-\sum_{s \in \text{Obs}(S)} |\mu^e_{\pi_n}(s) - \hat{\mu}_{\pi_n}(s)|}$. We generate a weight for each equilibrium behavior, and the vector of weights is then normalized to sum to 1.

Finally, the modified constraint of (10) is substituted with a weighted convex combination accounting for each equilibrium-based interaction. Let $\mathcal{E}$ be the number of equilibrium behaviors. Then, the constraint becomes:

$$\sum_{i=1}^{\mathcal{E}} \omega^{e_i} \cdot \sum_{s \in \text{Obs}(S)} \mu^{e_i}_{\pi_n}(s) \phi_k(s, \pi_n(s)) = \hat{\phi}_k \quad \forall k \quad (11)$$

During the IRL, as more observations are made, the weights are recomputed at each iteration. Initially the weights may be nearly uniform because the learned policies do not correctly model the observed behavior for the most part. Eventually, as the policies improve, projecting the true interaction behavior begins to matter and its corresponding weight increases compared to the others.

4. DOMAIN: PENETRATING A SIMPLE MULTI-ROBOT PATROL

We apply the approach described in Section 3 to the problem domain of two mobile robots, I and J, each executing cyclical patrolling trajectories in a common space independently for the most parts unless they approach each other from opposite directions. In order to continue moving past each other as smoothly as possible, the two robots must coordinate their actions. These events realize the interactions described in section 3.2 during which the joint behavior must be modeled. A subject robot, L, starting at a point in the space with a limited field of view – it can observe a portion of the patrolling robots' trajectories only – is tasked with autonomously reaching a goal state via a path that overlaps with the patrollers' trajectory, without being detected by any patroller. We show an example scenario in Fig. 3.

Each patroller is modeled as acting according to the output of its own MDP. The states of these MDPs are the cell decompositions of the common space, $\langle x, y \rangle$, along with the discretized orientation of the robot, ψ, and the actions of the patroller allow it to move forward one cell, stop, turn right or left 90° at its place or turn around 180° at its place. The transition function models the probability of any action failing at 33%. Subsequently, an unintended action is randomly selected for execution. Each patroller's reward function is modeled as a weighted linear combination of feature functions, as we mentioned in Section 2. As these pertain to the specific map used, we describe these in detail in the next section.

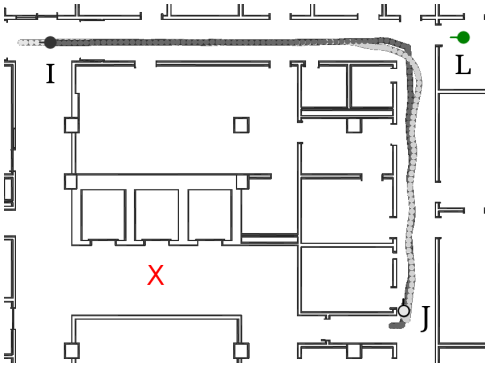

Figure 3: Trajectories of patrollers, I and J, in the hallways of a building. Subject robot L is tasked with reaching the goal state at X undetected from its starting position.

The subject robot shares the same state and action spaces as a patroller with one addition. To facilitate an accurate reward function, its state includes an additional variable, t, which is a time dimension discretized into steps as shown in Fig 4. The reward function of the subject robot assigns a positive number to a goal state regardless of the time step. However, modeling detections are challenging as the patrollers are mobile, which would make the reward function non-stationary. We may address this by considering the patrollers to be a part of the dynamic environment, and waiting until the patrollers have been observed and their policies learned. We may then jointly project their trajectories forward in time and space thereby indicating the location of each patroller at each time step. States at which the location of L is within visible distance of any patroller with both at the same time step are given a negative number. Each complete MDP is solved using the standard approach of value iteration [13].

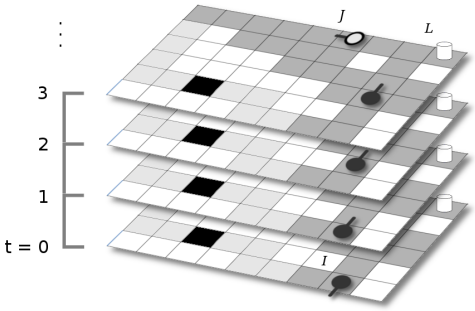

Figure 4: State space of the subject L (colored cells and time steps) including example predicted locations of I and J. The black cell is the goal and the white cells denote spaces that may not be traversed. I and J use a smaller physical state space (dark cells only) and without time.

An important open question so far is when the subject robot should start moving toward the goal? In order to answer this, we outline the algorithm utilized by the robot: As the patrollers may begin at any point in their trajectories, robot L stays at its starting location for a predefined time limit collecting observations of each patroller. Here, we assume that L is able to distinguish between the two patrollers and associate the observations with the patroller generating them. At the end of this phase, the gathered observations are utilized for learning each patroller's reward function using our approach for IRL under occlusion and with interaction modeling. The MDPs of each patroller are solved and its policy obtained.

Robot L then formulates its own MDP using the policies of the patrollers to project both their trajectories forward simultaneously from their most recent observations. The projections are utilized in formulating L's reward function, which completes its MDP. In Fig. 4, we show the state space of this MDP in the example map. States corresponding to the starting location of L at which the value as given by the value function is positive is indicative of finding a path from that state that leads to the goal without being detected. Within the set of such states, we focus on those whose time step remains in the future as the time elapses. L must simply wait until the time corresponding to the time steps in one of those states with the largest positive value has elapsed in order to start moving. Of course, none of the states may obtain a positive value. Subsequently, the subject robot continues to observe and utilizes its new observations to iterate over the procedure.

5. EXPERIMENTS

To better understand and evaluate our approach's performance in the application domain, we performed several experiments both in simulation and on physical robots. We utilize two distinct environments, one considerably larger than the other, in the simulations and perform our experiments with the physical robots in the smaller environment.

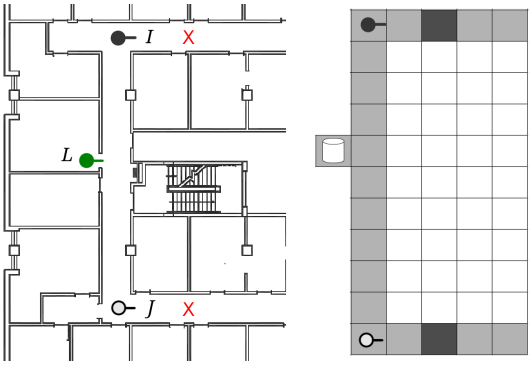

Figure 5: An environment and the corresponding MDP state space for our experiments. The two goal cells are colored black and the white cells denote occupied locations. L's starting location is shown while I and J move in a cycle between the two goal cells.

Our first environment is a simple hallway with restrictive routes and two possible goals (Fig. 5). Two robots, I and J, are known to be patrolling the hallways. They use simple cyclic trajectories, which are unknown to the subject robot, L, that starts just inside a room looking out an open door. This limits its field of view. The second environment is a larger portion of an office building floor (Fig. 3). The subject robot L located in a room on the outer rim of a building desires to reach the elevators near the center of the floor. In between, there are two hallways as well as open rooms. The hallways are patrolled by two robots. While the latter environment admits few paths for L to reach a goal, the former exhibits several potentially successful routes through the map depending on the positions of the patrollers. In each, L is spotted if it is roughly within 3 cells of any patroller that faces it.

The interactions between the patrollers in each environment occur when the two robots approach each other in a hallway from opposite directions. Each robot could then come to a stop, turn left or right, turn back around or side step, which involves the robot slowing down and moving forward while bypassing the other robot.

These interactions are time-extended requiring more than a single time step to perform (and they are upper bounded at 3).

Each robot in our simulations and physical experiments is a **TurtleBot** equipped with a Microsoft XBox 360 Kinect, which provides a camera and a ranging sensor. The base is an iCreate, which has a simple differential drive. Each robot also possesses a laptop running ROS Fuerte on Ubuntu 12.04. A robot identifies another by detecting its unique color signature using CMVision's blob finder. We use ROS's default local motion planner to locally navigate each robot. Each robot localizes itself in a map using the adaptive Monte Carlo localization method available in ROS.

Reward feature functions In the smaller environment, both $R_I(s,a)$ and $R_J(s,a)$ are composed of the following two types of binary feature functions as their trajectories are simple cycles:

1. *Has moved*, which returns 1 if the action causes the patroller to leave its current location, otherwise 0; and
2. *Turn around at state, s*, which returns 1 if the robot turns around 180° at the location given by s, otherwise 0.

Reward functions of the patrollers in the larger environment additionally include the following binary feature functions as their trajectories may go through rooms in the large hallways as well.

1. *Enter room*, which returns 1 if the patroller enters a room from one of the hallways, otherwise 0;
2. *In room*, which returns 1 if the patroller is in any of the rooms in the map, otherwise 0; and
3. *Leave room*, which returns 1 if the patroller leaves a room to enter a hallway.

The true vector of weights, $\boldsymbol{\theta}$, rewards moving within hallways and turning around after I or J has turned the corner. It penalizes turning around at locations within hallways. In the larger environment, it penalizes entering any side rooms from the hallways. Consequently, the MDP-based policy of each patroller generates trajectories that move through the hallways turning around near the corners at the end of each.

Comparative approaches In addition to our approach, which extends IRL to observing multiple robots under occlusion and which could be interacting (labeled as mIRL*+Interaction), we utilize three other approaches as baselines for a comparative performance analysis. mIRL* involves the IRL extended to multiple robots under occlusion but does not model interactions; an *upper bound* that is given the policy of each patroller and how they interact (labeled as KnownPolicy), and a random approach, which ignores all observations choosing a random time to move L.

5.1 Performance Evaluation in Simulations

In order to evaluate the performance of the approaches, a straightforward metric is the *success rate*. This is the proportion of runs in which the robot L reaches a goal state without being spotted by any patroller. Notice that this metric comprehensively measures the performance of all aspects of the approach and has practical implications. It is indicative of the accuracy of the learned patroller policies, L's simulated prediction accuracy, and the ability of L to traverse a route within an allotted time. In order to focus the evaluation on the performance of our generalized IRL, we additionally measure the accuracy of the learned policies of both I and J and their learned interaction behavior. The *learned behavior accuracy* is the proportion of all states at which the actions prescribed by the inversely learned policies of both patrollers or the interaction equilibrium (if it applies) coincide with their actual actions.

At each interaction state, mIRL*+Interaction formulates a game, such as the one shown in Fig. 2, in which the actions of a profile that resolves the conflict each receive a payoff of 5. Actions that do not change the physical location of a robot receive a payoff of 0, while others receive payoffs in between. Note that affine transformations of these payoffs do not alter the solution of the game.

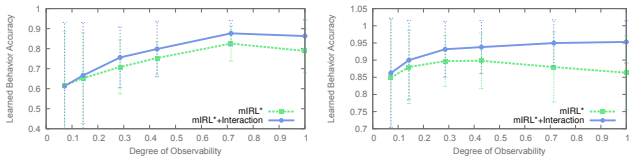

Figure 6: Learned behavior accuracy of our approach measured for different occlusion rates. The vertical bars represent one standard deviation from the mean.

Learned behavior accuracy We begin by evaluating the learned behavior accuracy of mIRL*+Interaction and mIRL* as a function of the degree of observability. The latter is the proportion of all (x,y) cells in the state space that are visible to the subject robot, L; its complement gives a measure of the occlusion. Note that KnownPolicy and random are irrelevant because these approaches do not learn the behavior of the other robots. Each data point is the average of 400 runs in the larger environment and 200 runs in the smaller one. Figure 6 shows this evaluation for both the simulated environments.

Observe that mIRL*+Interaction results in learning overall behavior of both I and J that is significantly more accurate in general compared to mIRL*, in both the environments (Student's paired, two-tailed t-test, $p < 0.025$ for both environments). The improvement becomes larger as the occlusion reduces. Clearly, modeling potential interactions in IRL matters as we may expect and leads to improved learning of others' behaviors. Furthermore, while the accuracy improves with the degree of observability for mIRL*+ Interaction, surprisingly it gradually reduces for mIRL* and at full observability drops down to the same level as when observability is at 10%. As the interactions between I and J become visible to L, mIRL* is unable to learn individual policies for the two robots that effectively model the joint behavior at the interaction states in their trajectories, likely attributing it to noise. On the other hand, mIRL*+ Interaction may utilize one of multiple equilibria to model those observations.

Success rate In Fig. 7, we show the varying success rates in both environments based on three important factors: the learned behavior accuracy, the degree of observability and the time L is given for observing the two patrolling robots. These are obtained for the same runs as before.

Improved accuracies of the learned behaviors has a positive impact on the success rate in the case of mIRL*+Interaction, as we show in Fig. 7(a). While both approaches attain the same level of accuracy – possibly due to better policy learning by mIRL* – the lack of interaction modeling by mIRL* leads to a reduced success rate. Note that our overall success rate does not exceed 60%, which is indicative of the general difficulty level of the problem that confronts L in this application domain.

Figures 7(b) and (c) show the impact of varying L's degree of observability and its observation time on the success rate, respectively. For each degree of observability, we evaluate over an identical distribution of observation times, and conversely for each observation time. Observe that KnownPolicy and random form

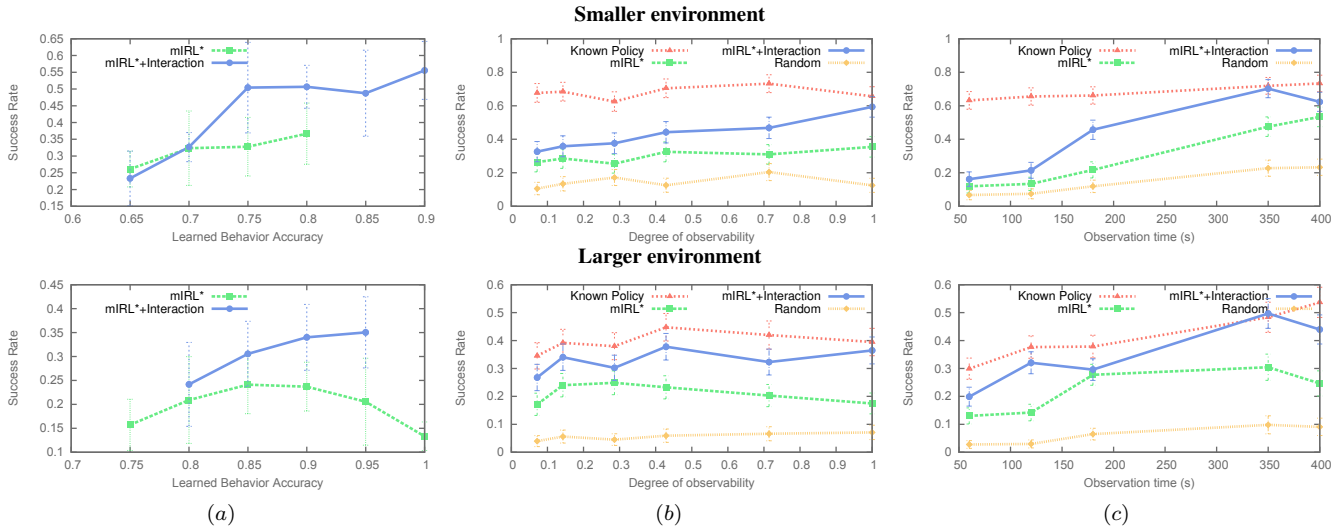

Figure 7: (a) **The effect of the learned joint behavior accuracy on the success rate in both simulated environments.** (b, c) **A comparison of the success rates achieved by our approach and the baselines as a function of the amount of visibility and the time spent observing. Vertical bars indicate 95% confidence intervals.**

the upper and lower bounds to the IRL-based approaches as expected. As the observability and time spent observing increases, mIRL*+Interaction's performance improves significantly. Indeed, for long observation times (e.g., 350s), the corresponding success rate matches that of the upper bound in both the environments. This implies that the generalized IRL is able to learn the observed behavior with a very high accuracy.

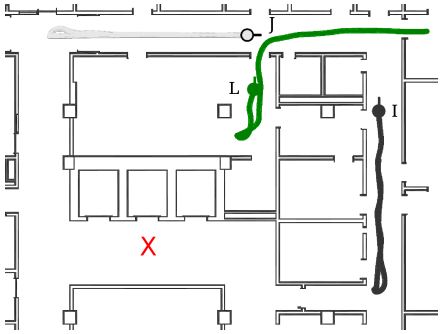

Figure 8: Trace showing a successful penetration in progress that involves the subject robot L entering a room as J passes by.

Finally, in Fig. 8 we show an interesting successful run by L, where its learning and subsequent MDP-based planning allows it to enter a side room to avoid being spotted as patroller J passes by it. Subsequently, it exits the room and successfully navigates to the goal location.

5.2 Evaluations on Physical Robots

We evaluate the performance of mIRL*+Interaction using physical robots. TurtleBots I and J physically patrol the hallway of the smaller environment (Fig. 5), while a third TurtleBot, L, is waiting in the side room observing the patrollers in its limited field of view. The latter is approximately equivalent to a 10% degree of observability as defined previously. The sensitivity of the blob finder on each robot is calibrated so that the detection distance approximately matches that utilized in the simulations. Each run has a time limit of 20 minutes with each action taking about 3.5s, after which an

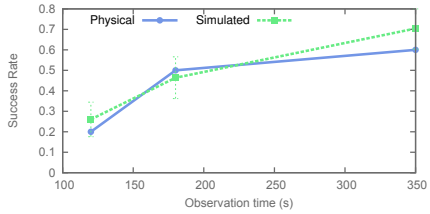

Figure 9: Success rates based on observation times in the physical runs. We compare these with those obtained from the simulations for the same degree of observability. The vertical bars are 95% confidence intervals.

incomplete run is aborted. Patrolling trajectories, feature functions and the MDP models of all the robots are identical to simulations.

As it was challenging to vary the degree of observability without L being immediately spotted, we show the success rates obtained by L in the physical runs when its observation time varies. In Fig. 9, we report the physical evaluations and compare it with the results from simulations for the same environment with a 10% degree of observability. The overall success rate starts low but improves to about 60%. Importantly, the difference is not statistically significant.

Figure 10 shows snapshots of robots I and J at an interaction state. Observe that I is stationary as J navigates around it. In addition, we show snapshots of L exiting its vantage point and moving along a path that successfully results in reaching the goal state.

6. RELATED WORK

Ng and Russell [11] introduced IRL as a problem involving a single subject agent learning from a single expert. We view IRL as a way of performing inverse optimal control [12] that allows for control frameworks other than MDPs as well. IRL lends itself naturally to a robot learning from demonstrations by one human teacher or expert in a controlled environment. Improvements and robotic applications include modeling the reward function as a linear combination of features [1], robot path planning [14], and learning with

Figure 10: (Top) I and J interacting, one stops movement while the other moves past at a reduced speed. (Bottom) L observing a patroller and performing a successful penetration

noisy feature functions [5]. However, these advances do not involve performing IRL in the presence of multiple experts; a significant difference and the primary motivation behind this paper.

Previous work on multiple robots involved in learning from demonstrations either as the learners or as the experts is sparse. One such domain is the scenario where a single expert teaches a team of robots [7]. Here, the learners consider each other's actions to facilitate coordination while the expert acts alone. Other multi-robot learning from demonstrations places a robot as an expert and another as the learner [3]. A challenge here is that the learner cannot directly imitate the expert's actions due to kinematic differences.

Modeling of robot interactions has received attention recently with Spaan and Melo [15] utilizing game theory to model interactions as well as a component of the multiagent planning. Analogous to our setting, two robots plan their trajectories using individual MDPs and overlay it with a Markov game when conflicts arise such as when they must cross through a narrow corridor. This problem is that of multiagent planning and not inverse learning and our work may be viewed as the inverse of this problem where we attempt to learn which Nash equilibrium the two robots have chosen based on their observed actions. Valtazanos and Ramamoorthy [16] use demonstrations to learn a set of interaction actions, which may then be used by the learner robot when interacting with another. This allows it to predict the state that may result due to its actions, and utilize these to shape the actions of a second robot as a response to the learner's actions. Our work differs in that we focus on entire trajectories where the interaction is a (critical) part of the perceived motion, the learner cannot influence the actions of the experts in any way, and we seek to learn the behavior of multiple robots when they are not interacting as well.

Finally, in the context of our application domain, related research has predominantly focused on generating robot patrolling trajectories that are theoretically difficult to learn from observations by relying on randomized behavior [2]. This direction is motivated by building robots that could be deployed for perimeter patrolling, which is complementary to our focus on exploring state-of-the-art techniques for building robots that may learn *simple* patrolling behavior of multiple other robots.

7. DISCUSSION

In the context of mobile robots occlusion of the state space is usually unavoidable. Our solution is to model the observability when performing IRL in order to learn a policy that best matches available observations of the robots. Our experiments show that even with highly occluded spaces and limited observation times useful policies are learned by the subject robot.

We have shown that in an application domain involving two interacting robots, modeling their interaction using a game results in increased learned behavior accuracy, which implies a corresponding improvement in the quality of predictions. In this paper, this was empirically shown by demonstrating the increased success rate for the algorithm which modeled the interaction.

8. ACKNOWLEDGMENTS

We gratefully acknowledge support from a NSF CAREER grant, IIS-0845036, and an ONR grant, N000141310870.

9. REFERENCES

[1] P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1, 2004.

[2] N. Agmon, S. Kraus, and G. a. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345, May 2008.

[3] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Correspondence mapping induced state and action metrics for robotic imitation. *IEEE Trans. Syst. man, Cybern. Part B, Cybern.*, 37(2):299–307, Apr. 2007.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Rob. Auton. Syst.*, 57(5):469–483, May 2009.

[5] A. Boularias, O. Krömer, and J. Peters. Structured apprenticeship learning. *Mach. Learn. Knowl. Discov. Databases*, pages 227–242, 2012.

[6] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[7] S. Chernova and M. Veloso. Teaching multi-robot coordination using demonstration of communication and state sharing (short paper). In *AAMAS 2008*, pages 1183–1186, 2008.

[8] J. Choi and K.-e. Kim. Inverse Reinforcement Learning in Partially Observable Environments. In *IJCAI*, pages 1028–1033, 2009.

[9] K. Dvijotham and E. Todorov. Inverse optimal control with linearly-solvable MDPs. In *ICML*, pages 335–342, 2010.

[10] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[11] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000.

[12] R. Obermayer and F. A. Muckler. *On the inverse optimal control problem in manual control systems*, volume 208. NASA, 1965.

[13] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[14] N. D. Ratliff, J. A. Bagnell, and M. a. Zinkevich. Maximum Margin Planning. In *ICML*, pages 729–736, 2006.

[15] M. Spaan and F. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *AAMAS*, pages 525–532, 2008.

[16] A. Valtazanos and S. Ramamoorthy. Bayesian interaction shaping: learning to influence strategic interactions in mixed robotic domains. In *AAMAS*, pages 6–10, 2013.

[17] B. Ziebart and A. Maas. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.