A Hierarchical Bayesian Process for Inverse RL in Partially-Controlled Environments

Kenneth Bogert
Computer Science Dept., University of North Carolina
Asheville, NC 28804, USA
kbogert@unca.edu

Prashant Doshi
THINC Lab, University of Georgia
Athens, GA 30602, USA
pdoshi@uga.edu

ABSTRACT

Robots learning from observations in the real world may encounter objects or agents in the environment, other than the expert giving the demonstration, that cause nuisance observations. These confounding elements are typically removed in fully-controlled environments such as virtual simulations or lab settings. When complete removal is impossible the nuisance observations must be filtered out. However, identifying the sources of observations when large amounts of observations are made is difficult. To address this, we present a hierarchical Bayesian process that models both the expert's and the confounding elements' observations thereby explicitly modeling the diverse observations a robot may receive. We extend an existing inverse reinforcement learning algorithm originally designed to work under partial occlusion of the expert to consider the diverse and noisy observations. In a simulated robotic produce-sorting domain containing both occlusion and confounding elements, we demonstrate the model's effectiveness. In particular, our technique outperforms several other comparative methods, second only to having perfect knowledge of the subject's trajectory.

KEYWORDS

Cobots; Maximum entropy; Produce sorting; Uncertainty

ACM Reference Format:
Kenneth Bogert and Prashant Doshi. 2022. A Hierarchical Bayesian Process for Inverse RL in Partially-Controlled Environments. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

A known modality for imitation learning in robotics [15] is to employ a sensor suite to learn from observing (LfO) the expert. The sensed data constitutes a trajectory, usually modeled by a Markov decision process (MDP), which specifies the state of and action taken by the subject agent at successive time steps as it performs the task. Techniques such as classical machine vision and deep learning may be combined to automate this process for complex sensors such as depth cameras [22]. On acquiring the trajectories, they are used by a machine learning method such as inverse reinforcement learning (IRL) [2, 18], to learn a model of the subject's demonstration so that an apprentice robot may then perform the same task.

While our ultimate concern pertains to the deployed robot's performance, a significant amount of effort goes into developing the portion of the LfO pipeline used only during demonstrations. It is this portion which yields the trajectories that are used in the learning. Clearly, a challenge for the learning is the inherent noise present in the sensor stream. The vision may fail to uniquely identify the state or action of the subject, deployed sensors may fail temporarily while observing the task being performed, or the detection is obfuscated due to the presence of confounding elements in the environment. Confounding elements are objects in the background, other agents moving through the environment, or objects that cause occlusion of the expert. Either by strictly controlling the environment during demonstrations, repeating demonstrations multiple times, or manually editing noisy sensor streams, such practical issues are overlooked by the extant literature.

In this paper, we present a general hierarchical Bayesian process that models the uncertainty present in observations of the expert agent demonstrations in partially controlled, real-world situations to perform imitation learning. We define a partially-controlled environment as one which contains a finite and static set of confounding elements, which are known. Our model exploits the underlying incomplete MDP that is being learned to provide structure to noisy observation data and is integrated into a previous technique for IRL under occlusion [5, 7] that generalizes the classical maximum entropy optimization. We demonstrate our model's effectiveness in a formative Gridworld scenario and in a larger robotic produce-sorting experiment in which a number of confounding elements are present.

2 PRELIMINARIES

IRL connotes both the problem and method by which an agent learns goals and preferences of another agent that explain the latter's observed behavior [18, 21]. The observed subject agent E is often considered an "expert" in the performed task. To model the subject agent, it is assumed that the expert is executing an optimal policy based on a standard MDP $\langle S, A, T, R\rangle$. The learning agent L is assumed to exhibit perfect knowledge of the MDP parameters except of the reward function. Therefore, the learner's task is to infer a reward function that best explains the observed behavior of the expert under these assumptions. A *policy* is a function mapping each state to an action. It can be deterministic, $\pi : S \rightarrow A$ or stochastic, $\pi : S \rightarrow Pr(A)$. The value function $V^{\pi} : S \rightarrow \mathbb{R}$ gives the value of a state s as the long-term expected cumulative reward obtained from the state by following π. The value of a policy π from some initial state s_0 is an expectation, $V^{\pi}(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right]$, where $\gamma \in (0, 1)$ is a discount factor.

The problem of IRL is generally ill-posed because for any given behavior there are infinitely-many reward functions which may explain the behavior. Ng and Russell [18] initially approached the problem with linear programming inferring a reward function that

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

maximizes the difference between the value of the expert's optimal policy and the next best policy under the assumption that the expert's complete policy is available. Abbeel and Ng [1] relaxed this assumption using an algorithm in which the expert, E, provides a *demonstration* of the task performance instead of its policy. (Demonstrations may be seen as composed of simulations of the expert's optimal policy.) The reward function is modeled as a linear combination of $K > 0$ binary features, $\phi_k \colon S \times A \longrightarrow [0, 1]$, $k \in \{1, 2 \ldots K\}$. Each feature maps a state from the set of states, S, and an action from E's set of actions, A_E, to a value in {0,1}. Choosing appropriate feature functions is important, and, if these are not known to the learner, they can be learned from the data thereby diminishing the need for feature engineering [17].

The reward function for the expert, E, is then defined as $R(s, a) = \boldsymbol{\theta}^T \phi(s, a) = \sum_{k=1}^{K} \theta_k \cdot \phi_k(s, a)$, where θ_k are the *weights* in vector $\boldsymbol{\theta}$; let $\mathcal{R} = \mathbb{R}^{|S \times A|}$ be the continuous space of reward functions. The learner task is simplified to one of completing the reward function by finding an appropriate vector of weights so that the demonstrated behavior is optimal. The data for IRL is in the form of a demonstration, which is usually comprised of trajectories. A trajectory of finite length T assumed to be generated by the MDP attributed to the expert is defined as $\mathbb{X}^T = \{X | X = (\langle s, a \rangle_1, \langle s, a \rangle_2, \ldots, \langle s, a \rangle_T)\}, \forall s \in S, \forall a \in A\}$. A demonstration is some finite, non-empty collection of trajectories of varying lengths, $\mathcal{X} = \{X | X \in \mathbb{X}^T\}$.

As the expert's policy is unavailable to the learner, the vector of weights is found by comparing the feature expectations of all possible trajectories to feature counts empirically estimated from the expert's policy [30]. The expert's feature expectations are estimated using the average of feature values for all observed trajectories, $\hat{\phi}_k = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \sum_{\langle s,a \rangle_t \in X} \phi_k(\langle s, a \rangle_t)$. The learner completes the expert's MDP using the learned reward function and may solve it to obtain π. Due to the ill-posed nature of the problem, the learned reward function may not be directly compared to the true function, if known. Instead, a popular metric is the *inverse learning error* (ILE) [11] of a learned reward R obtained as ILE $= ||V^{\pi^E} - V^{\pi^L}||_1$, where V^{π^E} is the value function computed using the *true reward* and the expert's optimal policy, V^{π^L} is the value function using the true rewards and the optimal policy for the *learned* reward.

2.1 Maximum Entropy IRL

The max-margin approach of Abeel and Ng [1] introduces a bias into the learned reward function. Biases can help guide the search in ill-posed problems, but they may preclude other meaningful solutions. Consequently, this motivates methods that make the least assumptions. Toward this, Ziebart et al. [30] finds the distribution over the trajectories that exhibits the maximum entropy while matching the observed feature expectations. The following nonlinear program gives this distribution.

$$\begin{aligned} &\max_{\Delta} \left(-\sum\nolimits_{X \in \mathbb{X}} Pr(X) \; log \; Pr(X) \right) \\ &\textbf{subject to} \; \sum\nolimits_{X \in \mathbb{X}} Pr(X) = 1 \\ &E_{\mathbb{X}}[\phi_k] = \hat{\phi}_k \; \forall k \end{aligned} \tag{1}$$

Here, $E_{\mathbb{X}}[\phi_k] = \sum_{X \in \mathbb{X}} Pr(X) \; \sum_{\langle s,a \rangle_t \in X} \phi_k(\langle s, a \rangle_t)$ is the feature expectation. The problem reduces to finding $\boldsymbol{\theta}$, which parameterizes the exponential distribution that exhibits the highest likelihood: $Pr(X) \propto e^{\sum_{\langle s,a \rangle_t \in X} \boldsymbol{\theta}^T \phi(s,a)}$. Notice that the chances of an expert agent following a trajectory is proportional to the cumulative reward incurred along that path. The benefit of this approach is that distribution $Pr(X)$ makes no further assumptions beyond those which are needed to match its constraints and is maximally noncommittal to any one trajectory. As such, it is most generalizable by being the least wrong most often of all alternative distributions. A disadvantage is that it becomes intractable for long trajectories because the set of trajectories grows exponentially with timesteps. To address this, another formulation, MaxCausalEntIRL [29], finds stochastic policies with maximum entropy.

2.2 IRL under Occlusion: HiddenDataEM

Our motivating application involves a collaborative robotic arm sorting produce as they move down a conveyor belt in a processing shed. It learns how to sort by using its sensors to observe an expert worker over an extended period. However, the activity may not be fully observed due to various reasons such as other persons accidentally walking in front of the camera. Previous methods [6] denote this special case of incomplete observability where some states are fully hidden as *occlusion.* Subsequently, the trajectories gathered by the learner exhibit missing data associated with timesteps where the expert is in one of the occluded states. The empirical feature expectation of the expert $\hat{\phi}_k$ would exclude the occluded states (and actions in those states).

To ensure that the feature expectation constraint of IRL methods accounts for the missing data, recent approaches [5, 7] take an expectation over the missing data conditioned on the observations. Completing the missing data in this way allows the use of all states in the constraint and with it the Lagrangian dual's gradient as well. The nonlinear program of (1) is modified to account for the hidden data and its expectation.

Let Y be the observed portion of a trajectory, H is one way of completing the hidden portions of this trajectory, and $X = Y \cup H$. Treating H as a latent variable gives a new definition for the expert's empirical feature expectations:

$$\hat{\phi}^{H|Y}_{\boldsymbol{\theta},k} \triangleq \frac{1}{|\mathcal{Y}|} \sum_{Y \in \mathcal{Y}} \sum_{H \in \mathbb{H}} Pr(H|Y; \boldsymbol{\theta}) \sum_{t=1}^{T} \phi_k(\langle s, a \rangle_t) \tag{2}$$

where $\langle s, a \rangle_t \in Y \cup H$, $\mathcal{Y}$ is the set of all observed Y, $\mathbb{H}$ is the set of all possible hidden H that can complete a trajectory. The program in (1) is modified by replacing $\hat{\phi}_k$ with $\hat{\phi}^{H|Y}_{\boldsymbol{\theta},k}$. Notice that in the case of no occlusion $\mathbb{H}$ is empty and $\mathcal{X} = \mathcal{Y}$. Therefore $\hat{\phi}^{H|Y}_{\boldsymbol{\theta},k} = \hat{\phi}_k$ and this method reduces to (1). Thus, this approach generalizes the previous maximum entropy IRL method. However, the program becomes nonconvex due to the presence of $Pr(H|Y)$. As such, finding its optima by Lagrangian relaxation is not trivial. Wang et al. [26] suggests a log-linear approximation that casts the problem of finding the parameters of the distribution (reward weights) as a likelihood maximization that can be solved within the schema of expectation-maximization [13]. An application of this approach to the problem of IRL under occlusion yields a method labeled as HiddenDataEM, which consists of the following two steps (with more details in [7]):

E-step This step involves calculating Eq. 2 to arrive at $\hat{\phi}^{H|_Y,(t)}_{\boldsymbol{\theta},k}$, a conditional expectation of the K feature functions using the parameter $\boldsymbol{\theta}^{(t)}$ from the previous iteration. We may initialize the parameter vector randomly.

M-step In this step, the modified program is optimized by utilizing $\hat{\phi}^{H|_Y,(t)}_{\boldsymbol{\theta},k}$ from the E-step above as the expert's feature expectations to obtain $\boldsymbol{\theta}^{(t+1)}$. Now, optimizing the relaxed Lagrangian becomes easier. Unconstrained adaptive exponentiated gradient descent [23], a version of gradient descent in which the learned parameter is scaled using the exponent of the gradient, solves the program. This variant of the gradient descent exhibits improved worst-case loss bounds over the standard gradient descent and often converges faster.

As EM may converge to local minima, this process is repeated with random initial $\boldsymbol{\theta}$ and the solution with the maximum entropy is chosen as the final one.

3 CONTROLLED ENVIRONMENTS

HiddenDataEM assumes that trajectories are noiselessly observed with some data missing due to occlusion. Data acquired from a robot's sensor suite, however, is likely to be noisy and provides partial information about the subject. We aim to generalize the E-step of HiddenDataEM to environmental and sensing noise to obtain a distribution over possible demonstrations $\mathcal{X}$ given the observations.

We first discuss a simplification of our model. It applies to scenarios where the learner (or its operator) has full control over the environment such that there is no possibility of extraneous observations and the learner has perfect knowledge of the state and action of the subject agent at each timestep. *This is a popular assumption for IRL, often possible in simulations only, but not practicable in many real-world robotic contexts.*

We emphasize that the sensor data may be markedly different from the state-action trajectory data. For instance, suppose a video camera is used to record the expert and a machine vision algorithm such as SIFT processes the video stream and produces a stream of features, which we call observations. If this camera runs at 30 fps, potentially *hundreds to thousands of observations* are generated every second, with many duplicates. Now, let the MDP timestep correspond to one wall-clock second and the state and action sets be discrete. Then, let these observations be modeled by an observation function (sensor model) dependent on the subject's state and action variable, i.e., $O(s, a, \omega)$, where $\omega \in \Omega$ are sensed observations (features). We show this model graphically in Figure 1 (left). Of note is that there may be any number of observations at each timestep, N_t. This characteristic makes it different from a hidden Markov model in which the state and action of the expert are also hidden but which models a single observation at each timestep.

It is generally impractical to have complete knowledge of O due to the nature of these complex observations. Instead, we allow O to be sampled from a Dirichlet distribution with hyperparameters $\boldsymbol{\alpha}$. Let the likelihood distribution of Ω be multinomial, which makes the Dirichlet over O a conjugate prior and allows us to set $\boldsymbol{\alpha}$ to be the count of each observation seen per state and action.

Controlled environments, such as simulations, facilitate this procedure as the state and action of the subject may be perfectly known.

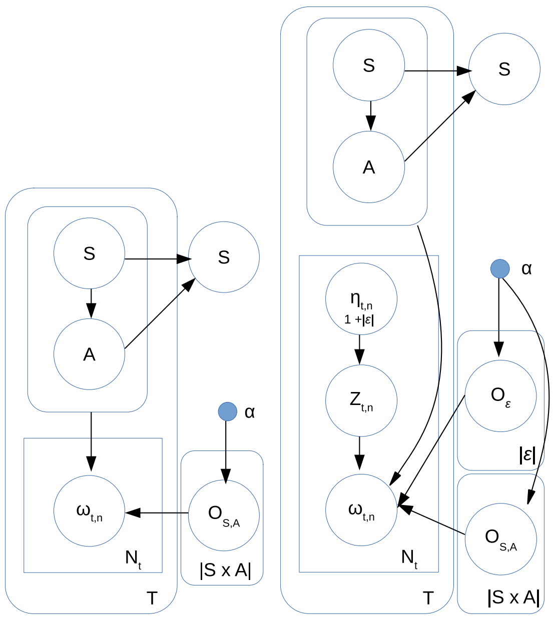

Figure 1: *(left)* A hierarchical Bayesian process for a controlled environment displayed in plate notation. At each timestep t, N_t observations $\boldsymbol{\omega}$ are sampled from the observation model $O_{S,A}$. *(right)* The partially-controlled environment requires an expanded hierarchical process model. Each observation $\omega_{t,n}$ is labeled with its source, $Z_{t,n}$, which may be the expert or a confounding element belonging to $\mathcal{E}$. As the true source may not be known, we sample it from the distribution $\eta_{t,n}$.

This learned model may then be transferred to less controlled environments. However, we may expect the expert to present some significant observation differences in this case. Therefore, we may need to scale the hyperparameters by a small constant to prevent overfitting.

4 PARTIALLY-CONTROLLED ENVIRONMENTS

Of course, LfO cannot usually be performed in sterile, controlled environments. Confounding elements often remain in the environment during demonstrations which cause extraneous observations. These observations must be filtered out in order to prevent incorrect trajectories from being produced that will distort the learning.

This is a problem of data association: if each observation has its source perfectly labeled we could simply exclude all non-subject agent observations and use the controlled environment model from Section 3. However, an observation's source may not be pinpointed perfectly due to the inherent noise. One common approach is to restrict the accepted observations in the hope that non-subject agent observations will be removed. But this increases the chances of occlusion when many of the available observations are not clear. This also precludes using unexpected, opportunistic observations, such as the presence of a mirror or a reflective surface in the environment, as these extra sources may get filtered out.

We take a novel approach with the goal of utilizing all available information while identifying and filtering out observations caused by confounding elements. Suppose we enumerate every confounding element present in the environment during demonstrations (this is reasonable assuming that the environment has been at least partially controlled to limit these elements). Then, the set of observations received at a given timestep is due to some mixture of these elements and the subject agent. Let $\mathcal{E}$ be the set of all confounding elements. To the previous Bayesian process model, we add a label variable Z for each observation ω, where Z identifies the source of ω and may take on a value either from $\mathcal{E}$ or one additional value indicating "subject agent". We expand the observation model to include all elements in $\mathcal{E}$. Note that when Z labels an ω as "subject agent" the observation model conditional on the subject agent's state and action is chosen, as was the case in our controlled environments model. Thus, Z acts as a multiplexer select input.

As machine vision is often uncertain about the identification of an observation, Z is itself unknown. *Thus, we model it as being sampled from a distribution $\eta(Z)$, which incorporates any information the observation system has about the source of a given ω and is of length $1 + |\mathcal{E}|$, with the first entry indicating "subject agent".* This allows for a rich set of information to be incorporated beyond simply that an ω was observed. An advanced system could track the movement of agents to help with the identification, it could interpret the observation in the context of others nearby, or express that an observation was vague or unusual. To illustrate, object detection using the Python ImageAI library could be employed on a RGB video stream to produce observations. These systems produce a probability distribution of possible object identifications for each detected object, which could be utilized as η. We show the new generalized hierarchical Bayesian process in Figure 1 (right).

4.1 Finding the Expert's Trajectory Distribution

Let $\mathbb{O} = \{\langle \boldsymbol{\omega}, \boldsymbol{\eta} \rangle_1, \langle \boldsymbol{\omega}, \boldsymbol{\eta} \rangle_2, \ldots, \langle \boldsymbol{\omega}, \boldsymbol{\eta} \rangle_{|\mathcal{X}|}\}$ be the set of observed trajectories produced from the subject's demonstration $\mathcal{X}$ each associated with a distribution over their corresponding labels (source distribution). To compute distributions over trajectories given α and $\mathbb{O}$ we also require the subject's policy $Pr(A|S)$. Unfortunately, as IRL is attempting to learn this distribution, we will not have the correct policy ahead of time. We resolve this by using the currently found trajectory distributions to compute $\hat{\phi}$ in the E-step of HiddenDataEM, revising Eq. 2 to yield Eq. 3. Using the policy produced by the subsequent M-step we iteratively improve our likelihood estimate of the subject's true demonstration.

$$\hat{\phi}^{\mathbb{O}}_{\boldsymbol{\theta},k} \triangleq \frac{1}{|\mathbb{O}|} \sum_{\langle \boldsymbol{\omega}, \boldsymbol{\eta} \rangle \in \mathbb{O}} \sum_{X \in \mathbb{X}} Pr(X|\boldsymbol{\omega}, \boldsymbol{\eta}; \boldsymbol{\theta}) \sum_{t=1}^{T} \phi_k(\langle s, a \rangle_t) \tag{3}$$

We may employ the Baum-Welch algorithm [19] to find the distributions $Pr(X|\boldsymbol{\omega}, \boldsymbol{\eta}; \boldsymbol{\theta})$. However, due to the large amounts of observation nodes per trajectory we replace the forward-backward message passing of Baum-Welch with Markov chain Monte Carlo sampling to improve performance. As shown in Algorithm 1 `Hierarchical Bayes Inference` present in Appendix A at the end of this paper, we first sample all the local nodes (S, A, Z) to produce distributions over all complete trajectories, update the observation model using these distributions, and repeat until convergence. We employ Metropolis-within-Gibbs sampling [24], a hybrid technique similar to Gibbs sampling, which samples individual nodes one at a time. *However, when the transition function of the underlying MDP is deterministic, we may not efficiently sample one state or action node at a time due to the probability of all transitions but one being zero.* Then, our algorithm samples a trajectory's entire set of state-action nodes at once using a method similar to Metropolis-Hastings.

4.2 Exploiting Indirect Observations

Although occlusion of the subject is one potential challenge in partially-controlled environments, it may be mitigated by exploiting *indirect observations* received from the subject.

By definition, the percept that causes an indirect observation does not travel directly from the subject agent to the learner's sensors, rather it reflects or bounces off of some part of the environment, appearing to come from elsewhere than the subject. These observations could include mirror reflections, natural pinspeck cameras [25], shadows [3], and changes in ambient color. Because they partly depend on the specific demonstration environment, these observations are difficult to manually specify and detect. It is also difficult to simulate them, as they rely on complex calculations such as ray tracing for optical physics, requiring extensive computation time.

As a simple instance of indirect observations countering occlusion, suppose a color blob finder is being used to identify an object being manipulated by a robot. If a reflective surface exists in the environment, such as a mirror behind the robot, we may receive *consistent* indirect observations from this surface. If the object is accidentally occluded from direct view, it is possible the reflection is still visible and can be used to identify the object.

Our approach is to allow the observation model, O, to be updated from the available demonstration data. This can be seen in line 23 of Algorithm 1 (see Appendix A at the end of this paper), where we gradually update α^i using $\dot{\alpha}$ found using the expected trajectories at iteration i and the previous values. In occluded timesteps, the number of observations directly caused by the subject agent is small, if any. Thus, any indirect observations greatly influence the resulting distribution over the expert's states and actions in these timesteps.

5 EXPERIMENTS

We implemented Algorithm 1 and evaluate it on a toy Gridworld as well as in our domain of key interest – the sorting of produce using a robotic manipulator.

5.1 Formative Tests on Gridworld

We evaluate our model first on a 5×5 Gridworld MDP. The Gridworld's reward function has four features corresponding to being in each of the four corners, with one corner randomly chosen to be the goal. We define four observations corresponding to each of the corners. The true observation function has each occurring exponentially more likely as the agent gets closer to the respective corner. For more details about the Gridworld, see Appendix B in the supplement.

To this model we add some confounding elements, each with its own random observation function. For our experiment we simulate the subject agent moving through the Gridworld 20 times, and at each timestep produces 40 observations sampled with equal probabilities from the subject agent's and each confounding element's observation function. Thus, as the number of confounding elements increases, the proportion of observations attributed to the subject reduces thereby making this a difficult learning task.

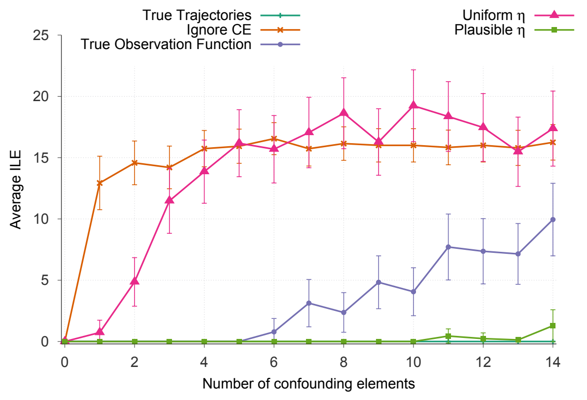

Figure 2: Average ILE of various techniques as the number of confounding elements increases in Gridworld. Each data point is the result of 65+ runs. Error bars are 95% confidence intervals on the mean. Lower ILE is preferred.

Figure 2 shows the ILE of the algorithm and several baselines. First, **True Trajectories** where the subject's true trajectories are input into MaxCausalEntropyIRL [28], has zero error as expected. Next is a variant where the **True Observation Function** O of the subject and those of the confounding elements are known to the learner but all η are uniform. Thus, the source Z of each observation is not known. It exhibits excellent performance with up to 6 confounding elements. In the next two variants, the observation model O is unknown but an informative Dirichlet prior α is used for the expert and a symmetric (uninformative) prior for the confounding elements.[1] **Uniform** η uses a uniform η distribution for all observations in the hierarchical Bayesian process while **Plausible** η has a 80% chance of assigning the true source of an observation 0.6 probability mass, otherwise a random source is given this mass. Finally, **Ignore CE** treats all observations as if they came from the subject and gives the lower bound. It achieves 0 error with no confounding elements but ILE rapidly rises as $|\mathcal{E}|$ grows. Overall, Fig. 2 shows that our technique with a plausible η performs nearly as well as having the true trajectories with up to 13 confounding elements.

5.2 Summative Tests using Robotic Onion Sorting

Domain description We now examine the performance of our technique in a larger produce sorting task. A simulated Sawyer robot arm is tasked with inspecting onions moving down a conveyor belt and sorting good onions from blemished ones. A blemished onion exhibits dark portions which may be visible while on the conveyor but will generally require a closer inspection by the robot's cameras. In this pick-inspect-place scenario we desire installing a new robot that will work alongside the existing one to increase sorting capacity. Rather than manually program the new robot, however, we wish to have it learn from watching the existing one's behavior in a production environment, such as a packing shed, where many confounding elements are present that cannot be removed. Examples of such elements include other shed workers and stations sorting produce other than onions. To assist the learner, we place a mirror behind the sorting robot to give the learner an additional viewpoint into the inspection. When the subject robot is occluded from the learner's view, *the mirror offers indirect observations of the onion being inspected and the expert's actions to the learner.*

A state of our MDP for the sorting task is factored and composed of 3 discrete variables: Onion Quality (unknown, good, blemished), Onion Position (on_conveyor, gripped, in_bin), and Gripper Position (conveyor, bin, inspection).

Actions cause certain changes in the state. Any action may be taken in any state but will have no effect (state unchanged) other than those described here.

- *Grip onion* changes an onion's position from on_conveyor to gripped with probability 1;
- *Release onion* changes an onion's position from gripped to in_bin or on_conveyor (based on the gripper's position) with probability 1;
- *Inspect onion* changes an onion's quality from unknown to either good or blemished by closely showing all sides of the onion to the robot's camera (only applicable in the inspection gripper position);
- *Move to conveyor*, *Move to inspection*, *Move to bin* changes the gripper's position to conveyor, inspection, and the bin each with probability 1, respectively.

Four general binary **reward features** are defined below and take the value 0 in all states and actions except in those described where they are valued 1.

(1) Release good onion in bin;
(2) Release good onion back on the conveyor belt;
(3) Release blemished onion in bin;
(4) Release blemished onion back on the conveyor belt.

The MDP assumes the subject is placing each type of onion either in the bin or back on the conveyor. *The learner is unaware of which action is chosen for which onion type.*

Trajectories are short (5-6 timesteps long) and terminate when Sawyer performs a *Release onion* action, thereby completing a single sort. Timesteps correspond to 2 secs of wall-clock time and we assume the learner is able to detect the beginning and end of trajectories (or an operator indicates them).

Our learner robot is equipped with a fixed RGB camera that is observing the expert. Determining the subject's complete state is not straightforward: Consider a blemished onion as shown in Fig. 3(a). It appears identical to a good onion from one angle, but from others has detectable dark spots. These appear to the learner as a distribution of discrete observations: bright onion and dark

[1] Informative observation model priors for confounding elements did not significantly impact performance in our experiments.

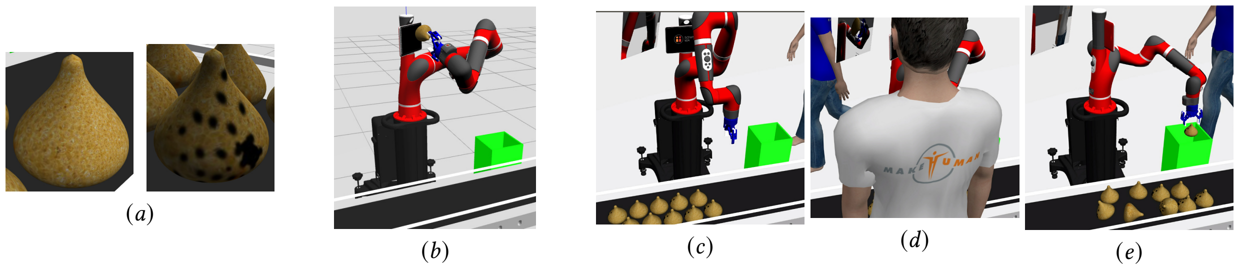

Figure 3: (a) A blemished onion from two viewpoints. (b) The onion sorting robot inspecting a bad onion in a fully controlled environment. (c,d,e) The same robot in a partially controlled environment sorting onions. Note the blue gripper and an individual's confounding blue shirt, occlusion due to a worker temporarily in front of camera, presence of other onions, and the mirror which provides the indirect observations (see the simulation video at https://youtu.be/r6IGU21Rty8 for more details).

onion, with blemished onions producing relatively more dark onion observations.

Additionally, confounding elements are present in the partially-controlled trials. These include moving persons in the environment that cause false positive observations or occlude the subject from view, and other onions not being sorted (see Fig. 3(c, d, e) and the *supplemental video*). The observations are defined below with more details in Appendix B in the supplementary material:

- Gripper blobs - The gripper is detected by running a color blob finder on the RGB frames. Gripper positions produce distributions of color blobs which are classified into one of 20 discrete observations each corresponding to an equal-sized region of the camera's f.o.v. The η for these observations is based on the size of the blob, with more mass given to the subject as the blob size approaches the bounding box size of the gripper in pixels. The leftover mass is attributed to a person walking behind the expert wearing a shirt of the same color as the gripper.
- Bright or Dark onion - Onions are detected in the RGB images using a color blob finder. The luminosity of the pixels constituting the onion is then classified as either bright or dark based on the proportion of dark pixels present. η is set to 80% for subject if the blob is proximal to a gripper blob, with the remainder going to other onions and a foreground person. If no gripper blob is seen nearby, we assign 60% mass to the other onions for small blobs or a foreground person for large ones, with the remainder divided between the subject (10%) and other confounding element (25%).

Notice that these observations do not unambiguously specify the state and action of the expert. The η values chosen represent our uncertain estimate of the source of the detected blobs, illustrating the difficulty of filtering out observations of confounding elements.

Experiment procedure We simulate the sorting robot with all confounding elements removed and record (i) the true state and action in each timestep and (ii) all observations received. We use these to find the α hyperparameters, as well as provide a ground truth trajectory for upper-bound purposes. Next, we repeat the same sort but with a mirror installed and confounding elements added, recording all observations. We produce the following data sets:

- Maximum likelihood (ML) trajectories - given the observation model and observations received we choose the most likely expert state and action at each timestep. This approximates common practice and serves as a baseline.
- ML observations - given η values as defined above, choose the observations that are most likely caused by the subject and use these observations only in the "controlled environment". This demonstrates the accuracy of η distributions.
- Partially-controlled model - we define $\mathcal{E}$ manually by reviewing the recorded demonstration and observation stream and listing all objects other than the subject that cause observations. We use uninformative hyperparameters α for all confounding elements, and use the observations in our partially controlled model. η is either **Plausible** (set to values described previously) or is **Uniform**.

We utilize MaxCausalEntIRL to find a reward function for data sets with known states and actions. For all others, we use HiddenDataEM and report metrics after Algorithm 1 has converged. We show the ILE produced by using the learned reward function to complete the given MDP versus using the true reward function for each data source. There were three confounding elements, which are evident in Fig. 3(c, d).

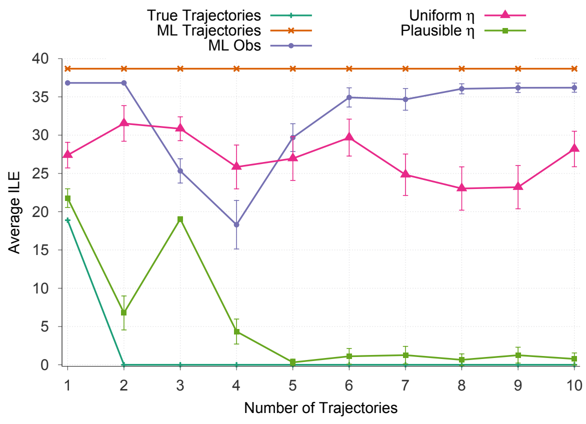

Figure 4: Average ILE in the onion sorting experiment as the number of trajectories increases. Each data point is from 100+ runs. Error bars are 95% confidence intervals on the mean.

As Fig. 4 reveals, two trajectories are needed for **True Trajectories** to achieve zero error due to the need to see at least one good and one blemished onion being sorted. **ML trajectories** is mostly flat because observations from the confounding elements prevent the likelihood estimation from yielding any reasonable trajectories. **ML Obs** shows sensitivity to the trajectories due to the variance in observations produced by each trajectory. **Plausible** η achieves very low error with 4 or more trajectories, while all other techniques fail to reduce error as trajectories increase. As a verification, on applying these techniques to observations produced in the fully controlled scenario, all methods achieve zero error with just two trajectories. Thus, HiddenDataEM [7] when integrated with the hierarchical Bayesian process and a plausible distribution over the sources of the learner's observations exhibits IRL under much uncertainty.

6 RELATED WORK

Bogert and Doshi [4, 6] introduce the challenges of occlusion that manifest in real-world applications of IRL to robotics. This extension of the original problem of IRL – which assumes full and perfect observability of the trajectories – is formally defined in a recent survey of the methods and progress in IRL [2]. An initial method [4] extended MaxEnt [30] to settings involving occlusion of portions of the trajectory by limiting the constraints to the observed portions only. Subsequently, the EM based approach reviewed in Section 2 was introduced, which forms an expectation over the missing data to allow the use of the Lagrangian gradient. However, this method suffers from a compute intensive expectation step when the occluded portions of the trajectories are long and contiguous. This limitation is addressed by a follow-up paper [5], which shows how blocked Gibbs sampling can be utilized to speed up the forward-backward message passing in the expectation step to allow scaling under occlusion to multiple experts. However, none of these generalizations to occlusion integrate indirect observations in the model as we do in this paper, which represents a different approach to manage occlusion.

On the other hand, techniques like BIRL [27], D-REX [8], and the more recent SSRR [9] target noisy trajectories due to the expert's failures during task performance. The first is based on the premise that noisy execution may cause the expert to sometimes exhibit off-policy actions. A latent variable characterizing the reliability of the action is introduced and an EM schema in the framework of BIRL manages this noise. The second technique (D-REX) automatically ranks demonstrated trajectories based on their total rewards using the Luce-Shepard rule while SSRR uses Adversarial IRL and assumes that the demonstrator is suboptimal and that pairwise preferences over trajectories are additionally needed for IRL. However, none of these methods introduce an observation model for the learner or account for partially occluded trajectories. As the expert in our setting fully and perfectly observes its state while the learner experiences noise due to imperfect sensors, IRL methods that model the expert as a partially observable MDP (POMDP) [11] are not related.

Kitani et. al. [16] introduce IRL with a single observation per timestep of the subject agent. However, the observation is modeled as caused by the subject's state only and no mechanism for exploiting the learned policy to improve the state distribution is developed. Choi and Kim [10] generalized Bayesian IRL [20] to a hierarchical model by introducing a prior over the temperature parameter that determines the randomness of the softmax distribution used in Bayesian IRL, and a hyperprior over the reward function prior. Maximum a-posteriori inference for Bayesian IRL [12] is then extended to the hierarchical version with a demonstration on the Gridworld and taxi problems [30]. A different hierarchical Bayesian model was also introduced by Dimitrakakis and Rothkopf [14] to generalize Bayesian IRL to learning multiple reward functions for multiple demonstrations. It involves a hyperprior on a joint reward-policy distribution for each of the demonstrated tasks. Posterior distributions are obtained using Metropolis-Hastings based sampling, which is evaluated on simple random MDPs. While these two methods involve the use of hierarchical models, these are relatively shallow compared to the final model we utilize here (Fig. 1(right)) and are not immediately generalizable to the challenges of noisy observations.

7 CONCLUDING REMARKS

We relax the strong assumption pervading IRL that the expert's trajectories are perfectly known to the learner. We introduce a novel hierarchical Bayesian process to model noisy observations whose inference is integrated into a previous IRL method that generalizes the classical maximum entropy technique. As demonstrated in our experiments involving a robotic domain, we successfully generalize IRL to realistic environments where confounding elements introduce noisy observations. The method is shown to be resilient to error in the source distribution η. Future work will explore the technique's sensitivity to η, allow this distribution to be learned from data, and extend the technique to uncontrolled scenarios where the set of confounding elements is unknown.

ACKNOWLEDGMENTS

This research is supported in part by NSF grant IIS-1830421 (to KB and PD), and a Phase 1 grant from the GA Research Alliance (to PD). We thank the anonymous reviewers for their valuable feedback.

A APPENDIX: ALGORITHM

Our main algorithm `Hierarchical Bayes Inference` is shown in Algorithm 1. The initialization procedure, not shown, involves finding an initial non-zero probability trajectory and sampling repeatedly to reduce the impact of this initial state on the final result.

Here, c is a small constant, σ is the number of desired samples, $Dirichlet\_mean()$ computes the mean distribution for a Dirichlet with the given hyperparameters, and the $SoftMaxPolicy(\boldsymbol{\theta})$ utilized in line 1 is defined as follows:

$$SoftMaxPolicy(\boldsymbol{\theta}) \triangleq Pr(a|s) \ = \ e^{(Q^{soft}(s,a;\boldsymbol{\theta}) - V^{soft}(s;\boldsymbol{\theta}))} \quad \forall\, s, a$$

where $Q^{soft}(s_t, a_t; \boldsymbol{\theta}) \ = \ E_{s_{t+1}}\left[V^{soft}(s_{t+1}; \boldsymbol{\theta})|s_t, a_t\right]$ and

$$V^{soft}(s_t; \boldsymbol{\theta}) \ = \ \text{softmax}_{a_t}\left[Q^{soft}(s_t, a_t) + \boldsymbol{\theta}^T \phi(s_t, a_t)\right].$$

Steps 12, 14, and 16 of Algorithm 1 involve sampling using Metropolis-within-Gibbs [24] and performed using Algorithm 2.

Algorithm 1: Hierarchical Bayes Inference

```
Input :$\mathbb{O}, \alpha^0, \theta$
Result: $Pr(X|\boldsymbol{\omega}, \boldsymbol{\eta})$
$Pr(a|s) \leftarrow SoftMaxPolicy(\theta) \quad \forall s, a$
$O_{s,a} \leftarrow Dirichlet\_mean(\alpha^0_{s,a}) \quad \forall\, s, a$
$O_Z \leftarrow Dirichlet\_mean(\alpha^0_Z) \quad \forall\, Z_{/1}$
$i \leftarrow 0$
while Not all O converged do
    $i \leftarrow i + 1$
    for $\boldsymbol{\omega}, \boldsymbol{\eta} \in \mathbb{O}$ do
        Initialize $\{ Z^0_{t,n} : t = 1, ...T, n = 1...N^t\}, \{ s^0_t : t = 1, ...T\}, \{ a^0_t : t = 1, ...T\}$
        for $j \leftarrow 1$ to $\sigma$ do
            for $t \leftarrow 1$ to $T$ do
                for $n \leftarrow 1$ to $N^t$ do
                    Sample $Z^{(j)}_{t,n} \sim Pr(Z_{t,n}|Z^{(j-1)}_{t,n}, \eta_{t,n}, \omega_{t,n}, O, s^{(j-1)}_t, a^{(j-1)}_t)$
                end
                Sample $s^{(j)}_t \sim Pr(s_t|s^{(j-1)}_t, a^{(j-1)}_t, s^{(j-1)}_{t+1}, s^{(j-1)}_{t-1}, a^{(j-1)}_{t-1}, \Omega_t, Z^{(j)}_t, O)$
                Sample $a^{(j)}_t \sim Pr(a_t|s^{(j)}_t, s^{(j)}_{t+1}, \Omega_t, Z^{(j)}_t, O)$
            end
        end
        Estimate $Pr(Z_{t,n}|\boldsymbol{\omega}, \boldsymbol{\eta})$ from $Z^{(1...\sigma)}_{t,n} \quad \forall\, Z, t, N^t$
        Estimate $Pr(X|\boldsymbol{\omega}, \boldsymbol{\eta})$ from $s^{(1...\sigma)}, a^{(1...\sigma)} \quad \forall\, X$
    end
    $\dot{\alpha}_{s,a} \leftarrow \sum_{\langle\boldsymbol{\omega},\boldsymbol{\eta}\rangle\in\mathbb{O}} \sum_{t\in\boldsymbol{\omega}} \sum_{n=1}^{|\boldsymbol{\omega}_t|} Pr(Z_{t,n} = 1|\boldsymbol{\omega}, \boldsymbol{\eta}) \sum_{X:s_t=s,a_t=a} Pr(X|\boldsymbol{\omega}, \boldsymbol{\eta}) \quad \forall\, s, a$
    $\dot{\alpha}_Z \leftarrow \sum_{\langle\boldsymbol{\omega},\boldsymbol{\eta}\rangle\in\mathbb{O}} \sum_{t\in\boldsymbol{\omega}} \sum_{n=1}^{|\boldsymbol{\omega}_t|} Pr(Z_{t,n} = Z|\boldsymbol{\omega}, \boldsymbol{\eta}) \quad \forall\, Z_{/1}$
    $\alpha^i \leftarrow c\, \dot{\alpha} + (1 - c)\alpha^{i-1} \quad \forall\, \alpha$
    $O_{s,a} \leftarrow Dirichlet\_mean(\alpha^i_{s,a}) \quad \forall\, s, a$
    $O_Z \leftarrow Dirichlet\_mean(\alpha^i_Z) \quad \forall\, Z_{/1}$
end
```

Algorithm 2: Sample

```
Given $y^{\tau-1}, \zeta$ = all other current node values
Simulate $\tilde{y} \sim U(Y)$
Take $y^\tau = \begin{cases} \tilde{y} & \text{with probability } p \\ y^{\tau-1} & \text{with probability } 1 - p \end{cases}$
Where: $p = \min\left(1, \frac{Pr(\tilde{y} \mid \zeta)}{Pr(y^{\tau-1} \mid \zeta)}\right)$
```

The sampling probabilities of individual nodes (y in Algorithm 2) are defined as:

$$Pr(Z_{t,n}|\eta_{t,n}, \omega_{t,n}, O, s_t, a_t) = \begin{cases} \eta_{t,n}(Z_{t,n})O_{s_t,a_t}(\omega_{t,n}), & \text{if } Z_{t,n} = 1 \\ \eta_{t,n}(Z_{t,n})O_{Z_{t,n}}(\omega_{t,n}), & \text{otherwise} \end{cases}$$

$$Pr(s_t|a_t, s_{t+1}, s_{t-1}, a_{t-1}, \Omega_t, Z_t, O) \;=\; Pr(a_t|s_t)\, Pr(s_t|s_{t-1}, a_{t-1}) \times Pr(s_{t+1}|s_t, a_t) \prod\nolimits_{n:Z_{t,n}=1} O_{s_t,a_t}(\omega_{t,n}) \text{ and}$$

$$Pr(a_t|s_t, s_{t+1}, \Omega_t, Z_t, O) \;=\; Pr(a_t|s_t)\, Pr(s_{t+1}|s_t, a_t) \times \prod\nolimits_{n:Z_{t,n}=1} O_{s_t,a_t}(\omega_{t,n})$$

Implementation note: When a large number of observations are present in a timestep the multiplication of the Z nodes in $Pr(s_t|\cdot)$ and $Pr(a_t|\cdot)$ above will not be numerically stable. In this situation, we modify the computation of the ratio $Pr(\tilde{y})/Pr(y^{\tau-1})$ to use a sum of log probabilities, ie. for s_t: $exp(\cdots + \big(log\, O_{\tilde{s}_t,a_t}(\omega_{t,n}) - log\, O_{s_t^{\tau-1},a_t}(\omega_{t,n})\big) + \ldots) \;\forall\, n : Z_{t,n} = 1$

We give the complete definitions of the MDPs for the two domains, the observation variables, η distributions and α priors, and the input trajectories in **Appendix B**, which is available in the supplementary material. We also show a video of our simulation of the noisy onion sorting domain at https://youtu.be/r6IGU21Rty8.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *Twenty-first International Conference on Machine Learning (ICML)*. 1–8.

[2] Saurabh Arora and Prashant Doshi. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence* 297 (2021), 103500.

[3] Manel Baradad, Vickie Ye, Adam B. Yedidia, Frédo Durand, William T. Freeman, Gregory W. Wornell, and Antonio Torralba. 2018. Inferring Light Fields From Shadows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6267–6275.

[4] Kenneth Bogert and Prashant Doshi. 2014. Multi-robot Inverse Reinforcement Learning Under Occlusion with Interactions. In *International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS) (AAMAS '14)*. 173–180.

[5] Kenneth Bogert and Prashant Doshi. 2017. Scaling Expectation-Maximization for Inverse Reinforcement Learning to Multiple Robots Under Occlusion. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 522–529.

[6] Kenneth Bogert and Prashant Doshi. 2018. Multi-robot inverse reinforcement learning under occlusion with estimation of state transitions. *Artificial Intelligence* 263 (2018), 46 – 73.

[7] Kenneth Bogert, Jonathan Feng-Shun Lin, Prashant Doshi, and Dana Kulic. 2016. Expectation-Maximization for Inverse Reinforcement Learning with Hidden Data. In *International Joint Conference on Autonomous Agents and Multiagent Systems.* 1034–1042.

[8] Daniel S Brown, Wonjoon Goo, and Scott Niekum. 2020. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*. PMLR, 330–359.

[9] Letian Chen, Rohan Paleja, and Matthew Gombolay. 2020. Learning from sub-optimal demonstration via self-supervised reward regression. *arXiv preprint arXiv:2010.11723* (2020).

[10] J. Choi and K. Kim. 2015. Hierarchical Bayesian Inverse Reinforcement Learning. *IEEE Transactions on Cybernetics* 45, 4 (2015), 793–805.

[11] Jaedeug Choi and Kee-Eung Kim. 2011. Inverse Reinforcement Learning in Partially Observable Environments. *J. Mach. Learn. Res.* 12 (2011), 691–730.

[12] Jaedeug Choi and Kee-Eung Kim. 2013. Bayesian Nonparametric Feature Construction for Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (Beijing, China) *(IJCAI '13)*. AAAI Press, 1287–1293.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* 39 (1977), 1–38. Issue 1.

[14] Christos Dimitrakakis and Constantin A. Rothkopf. 2012. Bayesian Multitask Inverse Reinforcement Learning. In *9th European Conference on Recent Advances in Reinforcement Learning*. 273–284.

[15] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation Learning: A Survey of Learning Methods. *Comput. Surveys* 50, 2, Article 21 (April 2017), 35 pages.

[16] Kris M. Kitani, Brian D. Ziebart, J. Andrew Bagnell, and Martial Hebert. 2012. Activity forecasting. In *European Conference on Computer Vision (ECCV)*. 201–214.

[17] Sergey Levine, Zoran Popović, and Vladlen Koltun. 2010. Feature Construction for Inverse Reinforcement Learning. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems (NIPS)*. 1342–1350.

[18] Andrew Ng and Stuart Russell. 2000. Algorithms for inverse reinforcement learning. In *Seventeenth International Conference on Machine Learning*. 663–670.

[19] Lawrence Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*. 77(2):257–286.

[20] Deepak Ramachandran and Eyal Amir. 2007. Bayesian Inverse Reinforcement Learning. In *20th International Joint Conference on Artifical Intelligence (IJCAI)*. 2586–2591.

[21] Stuart Russell. 1998. Learning Agents for Uncertain Environments (Extended Abstract). In *Eleventh Annual Conference on Computational Learning Theory*. 101–103.

[22] Nihal Soans, Ehsan Asali, Yi Hong, and Prashant Doshi. 2020. SA-Net: Deep Neural Network for Robot Trajectory Recognition from RGB-D Streams. In *International Conference on Robotics and Automation (ICRA)*.

[23] Jacob Steinhardt and Percy Liang. 2014. Adaptivity and Optimism: An Improved Exponentiated Gradient Algorithm. In *31st International Conference on Machine Learning*. 1593–1601.

[24] Luke Tierney. 1994. Markov chains for exploring posterior distributions. *the Annals of Statistics* (1994), 1701–1728.

[25] Antonio Torralba and William T Freeman. 2012. Accidental pinhole and pinspeck cameras: Revealing the scene outside the picture. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 374–381.

[26] Shaojun Wang and Dale Schuurmans Yunxin Zhao. 2012. The Latent Maximum Entropy Principle. *ACM Transactions on Knowledge Discovery from Data* 6, 8 (2012).

[27] Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. 2014. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[28] Brian Ziebart. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Ph.D. Dissertation. Carnegie Mellon University.

[29] BD Ziebart, JA Bagnell, and AK Dey. 2010. Modeling interaction via the principle of maximum causal entropy. In *ICML*.

[30] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *23rd National Conference on Artificial Intelligence - Volume 3*. 1433–1438.