Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks

Stefanos Nikolaidis [*] Ramya Ramakrishnan
Keren Gu Julie Shah

snikol@alum.mit.edu, ramyaram@mit.edu, kgu@mit.edu, julie_a_shah@csail.mit.edu
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

ABSTRACT

We present a framework for automatically learning human user models from joint-action demonstrations that enables a robot to compute a robust policy for a collaborative task with a human. First, the demonstrated action sequences are clustered into different human types using an unsupervised learning algorithm. A reward function is then learned for each type through the employment of an inverse reinforcement learning algorithm. The learned model is then incorporated into a mixed-observability Markov decision process (MOMDP) formulation, wherein the human type is a partially observable variable. With this framework, we can infer online the human type of a new user that was not included in the training set, and can compute a policy for the robot that will be aligned to the preference of this user. In a human subject experiment ($n = 30$), participants agreed more strongly that the robot anticipated their actions when working with a robot incorporating the proposed framework ($p < 0.01$), compared to manually annotating robot actions. In trials where participants faced difficulty annotating the robot actions to complete the task, the proposed framework significantly improved team efficiency ($p < 0.01$). The robot incorporating the framework was also found to be more responsive to human actions compared to policies computed using a hand-coded reward function by a domain expert ($p < 0.01$). These results indicate that learning human user models from joint-action demonstrations and encoding them in a MOMDP formalism can support effective teaming in human-robot collaborative tasks.

1. INTRODUCTION

The development of new industrial robotic systems that operate in the same physical space as people highlights the emerging need for robots that can integrate seamlessly into human group dynamics. In order to be efficient and productive teammates, robotic assistants need to be able to adapt to the personalized style of their human counterparts. This adaptation requires learning a statistical model of human behavior and integrating this model into the decision-making algorithm of the robot in a principled way.

In this paper, we describe a framework that allows for the learning of human user models through joint-action demonstrations. This framework enables the robot to compute a robust policy for a collaborative task with a human, assuming access to demonstrations of human teams working on the task. We hypothesize that a limited number of "dominant" strategies can capture the majority of demonstrated sequences. Using this assumption, we denote the preference of a human team member as a partially observable variable in a mixed-observability Markov decision process (MOMDP) [24] and constrain its value to a limited set of possible assignments. We chose the MOMDP formulation because the number of observable variables for human-robot collaborative tasks in a manufacturing setting is much larger than that of partially observable variables. Denoting the human preference for the action toward task completion as a hidden variable naturally models human collaboration, since the intentions of the participants can never be directly observed during training, and must be inferred through interaction and observation.

We start by describing the clustering of demonstrated action sequences into different human types using an unsupervised learning algorithm. These demonstrated sequences are used by the robot to learn a reward function that is representative for each type, through the employment of an inverse reinforcement learning algorithm. The learned models are then incorporated as part of a MOMDP formulation. With this framework, we can infer, either offline or online, the human type of a new user that was not included in the training set, and can compute a policy for the robot that will be aligned to the preference of this new user. In a human subject experiment ($n = 30$), participants agreed more strongly that the robot anticipated their actions when working with a robot utilizing the proposed framework ($p < 0.01$), compared to manually annotating robot actions. Additionally, in trials where the sequence of robot actions toward task completion was not trivially inferred by the participants, the proposed framework significantly improved team efficiency

[*] Stefanos Nikolaidis' present affiliation is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

HRI'15, March 2–5, 2015, Portland, Oregon, USA.
Copyright © 2015 ACM 978-1-4503-2883-8/15/03 ...$15.00.
http://dx.doi.org/10.1145/2696454.2696455.

($p < 0.01$). Finally, the robot was found to be more responsive to human actions with the learned policy, compared with executing a policy from a reward function hand-coded by a domain expert ($p < 0.01$).

First, we place our work in the context of other related work in Section 2, and introduce the proposed framework in Section 3. Next, we describe the clustering of demonstrated action sequences in Section 4. The learned models are then used as part of a MOMDP formulation (Section 5). We describe the human subject experiment in Section 6, discuss the results in Section 7 and conclude with potential directions for future research in Section 8.

2. RELEVANT WORK

For a robot to learn a human model, a human expert is typically required to explicitly teach the robot a skill or specific task [3, 4, 1, 21, 8, 2]. In this work, demonstrations of human teams executing a task are used to automatically learn human types in an unsupervised fashion. The data from each cluster then serves as input for an inverse reinforcement learning algorithm. In the context of control theory, this problem is known as inverse optimal control, originally posed by Kalman and solved in [6]. We follow the approach of Abbeel and Ng [1], and solve a quadratic program iteratively to find feature weights that attempt to match the expected feature counts of the resulting policy with those of the expert demonstrations. There have also been game-theoretic approaches [28, 30] that aim to model multi-agent behavior. Recent state-of-the-art imitation learning algorithms [15] have been shown to preserve theoretical guarantees of performance, while minimizing risk during learning. Through human demonstrations, our framework learns a number of different human types and a reward function for each type, and uses these as part of a MOMDP formulation.

Related approaches to learning user models include natural language interaction with a robot wheelchair [9], where a user model is learned simultaneously with a dialog manager policy. In this case, the system assumes that the model parameters are initially uncertain, and improves the model through interaction. More recently, humans and robots have been able to learn a shared plan for a collaborative task through a training process called "cross-training" [22]. Such approaches do not have the limitation of a fixed set of available models; however, learning a good model requires a large amount of data, which can be an issue when using the model for large-scale, real-world applications. Rather than learning a new model for each human user, which can be tedious and time-consuming, we use demonstrations by human teams to infer some "dominant" human types, and then associate each new user to a new type.

Recent work has also inferred human intentions during collaborative tasks for game AI applications. One such study [20] focused on inferring the intentions of a human player, allowing a non-player character (NPC) to assist the human. Alternatively, [17] proposed the partially observable Monte-Carlo cooperative planning system, in which human intention is inferred for a "cops-and-robbers" turn-based game. In both works, the model of the human type is assumed to be known beforehand.

Partially observable Markov decision process (POMDP) models have been used to infer human intention during driving tasks [7], as well. In this case, the user model is represented by the transition matrix of a POMDP and is learned through task-specific action-rules. Recently, the mixed observability predictive state representation framework (MO-PSR) [23] has been shown to learn accurate models of mixed observability systems directly from observable quantities, but has not yet been verified for task planning applications. The MOMDP formulation [24] has been shown to achieve significant computational efficiency, and has been used in motion planning applications [5], with uncertainty about the intention of the human over their own actions. In the aforementioned work, the reward structure of the task is assumed to be known. In our work, the reward function that corresponds to each human type is learned automatically from unlabeled demonstrations.

In summary, we present a pipeline to automatically learn the reward function of the MOMDP through unsupervised learning and inverse reinforcement learning. The proposed framework enables the rapid estimation of a human user model online, through the a priori unsupervised learning of a set of "dominant models" encoded in a MOMDP formulation. Using a human subject experiment, we show that the learned MOMDP policies accurately represent the preferences of human participants and can support effective teaming for human-robot collaborative tasks. We describe the proposed framework in the next section.

3. METHOD

Our framework has two main stages, as shown in Figure 1. The training data is preprocessed in the first stage. In the second stage, the robot infers the personalized style of a new human teammate and executes its role in the task according to the preference of this teammate.

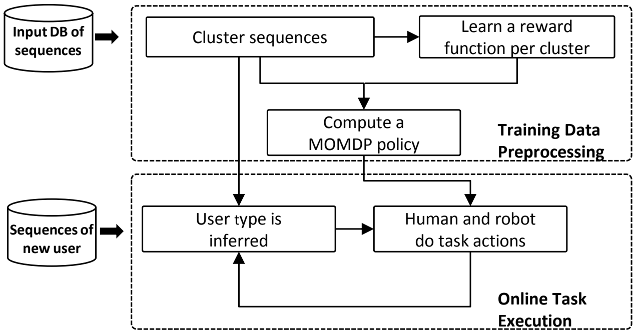


Figure 1: Framework flowchart

When a robot is introduced to work with a new human worker, it needs to infer the human type and choose actions aligned to the preference of that human. Additionally, the robot should reason over the uncertainty on the type of the human. The first stage of our framework assumes access to a set of demonstrated sequences of actions from human teams working together on a collaborative task, and uses an unsupervised learning algorithm to cluster the data into dominating human types. The cluster indices serve as the values of a partially observable variable denoting human type in a mixed-observability Markov decision process. Our framework then learns a reward function for each human type, which represents the preference of a human of the given type on a subset of task-related robot actions. Finally, the framework computes an approximately optimal policy for the robot that reasons over the uncertainty on the human type and maximizes the expected accumulated reward.

In the second stage, a new human subject is asked to execute the collaborative task with the robot. The human can then demonstrate a few sequences of human and robot actions, and a belief about his type can be computed according to the likelihood of the human sequences belonging to each cluster. Alternatively, if the human actions are informative of his type, as in the human subject experiment described in Section 6, the human type can be estimated online. The robot then executes the action of the computed policy of the MOMDP, based on the current belief of the human type at each time step.

In the following section, we describe the first block of the proposed framework: finding the number of dominating human types in a collaborative task by clustering the demonstrated sequences.

4. CLUSTERING OF HUMAN TYPES

To improve a robot's ability to adapt to human preferences, we first try to find human preferences using an unsupervised clustering approach. In this problem, we have a data set $D = x_1, ..., x_n$, where each x_i is a demonstrated sequence of alternating, discrete human and robot actions. The robot actions are those that the human annotates for the robot. The goal is to find the number of human types, k, within this data and the assignment of each sequence of actions x_i to a type.

Previous work has approached this problem of clustering sequential data through various methods. Murphy and Martin [19] clustered ranking or ordinal data through expectation-maximization (EM) by learning distance-based models that had two parameters: a central ranking and a precision parameter. The distance between rankings was defined using Kendall's, Spearman's and Cayley's distances, as specified in [18]. In another work, Jääskinen [13] clustered DNA sequences modeled as Markov chains using a Dirichlet process prior over the partitions. A greedy search of joining and splitting partitions was used to determine the number of clusters, and EM was used to learn transition probability matrices and to correctly assign sequences to clusters. In solving our clustering problem, we chose to use a hybrid approach combining these two methods. Similar to [13], our framework learns transition matrices between human and robot actions using EM, because this provides information about how the human will act based on the actions of the robot, and vice-versa.

We begin by using a hard variant of EM, similar to [13], to cluster the data into a set of human preferences. In the algorithm, we represent each preference or cluster by a transition matrix of size $|A|$ x $|A|$, where $|A|$ is the size of the action space, $A = \{A_r, A_h\}$, which includes both robot actions A_r and human actions A_h. Since the data consists of a sequence of actions in which the human and robot take turns, the transition matrix encodes information about how the human will act based on the previous robot action, and vice-versa. We then define $\boldsymbol{\theta}$ as the set of k representative transition matrices $\theta_1, ..., \theta_k$ that correspond to the k clusters. Every sequence x_i, each of length l, in the data $D = x_1...x_n$ must be assigned to one of these k clusters. The assignments of these sequences to clusters can be denoted as $Z = z_1...z_n$, where each $z_i \in \{1, ..., k\}$.

Algorithm: Cluster-Transition-Matrices (k)

```
1. Initialize $\hat{\boldsymbol{\theta}}$ by randomizing $\hat{\theta}_1, ..., \hat{\theta}_k$
2. Initialize sequence assignments $Z = z_1, ..., z_n$
3. repeat
4.    E-step: Compute assignments for each sequence $z_i$
      for $i = 1, ..., n$
      $z_i = \arg\max_{z_i} \left( \prod_{j=2}^{l} \hat{\theta}_{z_i}(x_i^j | x_i^{j-1}) \right)$
5.    M-step: Update each transition matrix $\hat{\theta}_z$
      for $z = 1, ..., k$
      $n_{i|j}$ : observed count of transitions from $i$ to $j$
      $\hat{\theta}_{z,i|j} = \frac{n_{i|j}}{\sum_{x=1}^{|A|} n_{x|j}}$ for $i, j = 1, ..., |A|$
6. until $Z$ converges to stable assignments
```

Figure 2: Cluster Transition Matrices using EM

The probability of one sequence x_i parameterized by $\boldsymbol{\theta}$ can be represented as follows, where x_i^j denotes the j^{th} element of the i^{th} demonstrated sequence:

$$\begin{aligned} P(x_i; \boldsymbol{\theta}) &= \sum_{z_i=1}^{k} P(z_i) P(x_i | z_i; \boldsymbol{\theta}) \\ &= \sum_{z_i=1}^{k} P(z_i) \left(\prod_{j=2}^{l} \theta_{z_i}(x_i^j | x_i^{j-1}) \right) \end{aligned} \tag{1}$$

For all data points, the log-likelihood is:

$$\begin{aligned} l(D; \boldsymbol{\theta}) &= \sum_{i=1}^{n} log P(x_i; \boldsymbol{\theta}) \\ &= \sum_{z=1}^{k} \sum_{i=1}^{n} \delta(z|z_i) log \left(P(z_i) \prod_{j=2}^{l} \theta_{z_i}(x_i^j | x_i^{j-1}) \right) \end{aligned} \tag{2}$$

$\delta(z|z_i) = 1$ if $z = z_i$ and zero otherwise.

The cluster-transition-matrices EM algorithm learns the optimal transition matrices $\hat{\theta}_1, ..., \hat{\theta}_k$ by iteratively performing the E-step and the M-step. First, lines 1-2 randomly initialize k transition matrices and sequence assignments; then, lines 3 through 6 repeatedly execute the E-step and M-step until the assignments Z converge to stable values. In the E-step, we complete the data by assigning each sequence to the cluster with the highest log-likelihood (line 4). In the M-step, each cluster's transition matrix is updated by counting the transitions in all sequences assigned to that cluster (line 5). These two steps are repeated until the assignments $z_1, ..., z_n$ do not change (line 6).

For the EM algorithm, we choose the number of clusters, k, by calculating the within-cluster dispersion for a range of values of k and selecting the value of k such that adding another cluster does not result in a significant decrease of the within-cluster dispersion [29].

We then input the learned clusters into a mixed-observability Markov decision process, as described in the next section.

5. MOMDP LEARNING AND PLANNING

The clusters of demonstrated action sequences represent different types of humans. When a robot is introduced to work with a new human worker, it needs to infer the human type for that worker and choose actions that are aligned to their preference. Additionally, the robot should reason over the uncertainty on the type of the human. Therefore, the cluster indices serve as the values of a partially observable variable denoting the human type in a mixed-observability Markov decision process (MOMDP).

Our framework learns a reward function for each human type. We then compute an approximately optimal policy for the robot that reasons over the uncertainty on the human type and maximizes the expected accumulated reward. We describe the MOMDP formulation, the learning of the reward function and the computation of an approximately optimal policy, as follows:

5.1 MOMDP Formulation

We treat the unknown human type as a hidden variable in a MOMDP, and have the robot choose actions according to the estimated human type. The MOMDP framework uses proper factorization of the observable and unobservable state variables, reducing the computational load. The MOMDP is described by a tuple, $\{X, Y, S, A_r, \mathcal{T}_x, \mathcal{T}_y, R, \Omega, O\}$, so that:

- X is the set of observable variables in the MOMDP. In our framework, the observable variable is the current task-step among a finite set of task-steps that signify progress toward task completion.
- Y is the set of partially observable variables in the MOMDP. In our framework, a partially observable variable, y, represents the human type.
- $S : X \times Y$ is the set of states in the MOMDP consisting of the observable and non-observable variables. The state $s \in S$ consists of the task-step x, which we assume is fully observable, and the unobservable type of the human y.
- A_r is a finite set of discrete task-level robot actions.
- $\mathcal{T}_x : S \times A_r \longrightarrow \Pi(X)$ is the probability of the fully observable variable being x' at the next time step if the robot takes action a_r at state s.
- $\mathcal{T}_y : S \times A_r \times X \longrightarrow \Pi(Y)$ is the probability of the partially observable variable being y' at the next time step if the robot takes action a_r at state s, and the next fully observable state variable has value x'.
- $R : S \times A_r \longrightarrow \mathbb{R}$ is a reward function that gives an immediate reward for the robot taking action a_r at state s. It is a function of the observable task-step x, the partially observable human type y and the robot action a_r.
- Ω is the set of observations that the robot receives through observation of the actions taken by the human and the robot.
- $O : S \times A_r \longrightarrow \Pi(\Omega)$ is the observation function, which gives a probability distribution over possible observations for each state s and robot action a_r. We write $O(s, a_r, o)$ for the probability that we receive observation o given s and a_r.

5.2 Belief-State Estimation

Based on the above, the belief update is then [24]:

$$b_y(y') = \eta O(s', a_r, o) \sum_{y \in Y} \mathcal{T}_x(s, a_r, x')\mathcal{T}_y(s, a_r, s')b_y(y) \quad (3)$$

5.3 Inverse Reinforcement Learning

Given a reward function, an exact value function and an optimal policy for the robot can be calculated. Since we want the robot to choose actions that align with the human type of its teammate, a reward function must be specified for every value that the human type can take. Manually specifying a reward function for practical applications can be tedious and time-consuming, and would represent a significant barrier for the applicability of the proposed framework. In this section, we describe the learning of a reward function for each human type using the demonstrated sequences that belong to the cluster associated with that specific type.

For a fixed human type y, we can reduce the MOMDP into a Markov decision process (MDP). The MDP, in this context, is a tuple: (X, A_r, T_x, R, γ), where X, A_r, T_x and R are defined in the MOMDP Formulation section above. Given demonstrated sequences of state-action pairs, we can estimate the reward function of the MDP using the inverse reinforcement learning (IRL) algorithm [1]. We assume the human type to be constant in the demonstrated sequences. To compute the reward function for each cluster, we first assume that a feature vector φ exists for each state, and each given policy has a feature expectation that represents the expected discounted accumulation of feature values based on that policy. Formally, we define the feature expectations of a policy π to be:

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \varphi(s_t)|\pi] \quad (4)$$

We also require an estimate of the feature expectations for each human type. Given a set of n_z demonstrated state-action trajectories per human type z, we denote the empirical estimate for the feature expectation as follows:

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i=1}^{n_z} \sum_{t=0}^{\infty} \gamma^t \varphi(s_t^{(i)}) \quad (5)$$

The IRL algorithm begins with a single random policy and attempts to generate a policy that is a mixture of existing policies, with feature expectations that are similar to those for the policy followed by the expert. In our case, the "expert" demonstrations were those followed by all humans of a particular type. The algorithm terminates when $||\hat{\mu_z} - \mu(\tilde{\pi})||_2 \leq \epsilon$, and is implemented as described in [1]. The output is a list of policies $\pi^{(i)}$ and corresponding reward functions $R^i(s)$, with mixture weights λ_i. We use the reward function of the policy with the maximum weight λ_i.

For each human type, the framework applies IRL, using the demonstrated sequences of that type as input to calculate an associated reward function. With a reward function for any assignment of the partially observable human type variable y, we can now compute an approximately optimal policy for the robot, as described in the next section.

5.4 Policy Computation

We solve the MOMDP for a policy that takes into account the uncertainty of the robot over the human type, while maximizing the agent's expected total reward. MOMDPs are structured variants of POMDPs, and finding an exact solution for a POMDP is computationally expensive [14]. Point-based approximation algorithms have greatly improved the speed of POMDP planning [27, 16, 26] by updating selected sets of belief points. In this work, we use the SARSOP solver [16], which, combined with the MOMDP formulation, can scale up to hundreds of thousands of states [5]. The SARSOP algorithm samples a representative set of points from the belief space that are reachable from the initial belief, and uses this set as an approximate representation of the space, allowing for the efficient computation of a satisfactory solution.

6. HUMAN-ROBOT TEAMING EXPERIMENT

We conducted a large-scale experiment to evaluate the proposed framework. In particular, we were interested in showing that the learned policies enabled the robot to take anticipatory actions that matched the preference of a human worker, compared to policies computed using a reward function hand-coded by a domain expert. We were also interested in the quality of the learned reward function compared to a hand-coded reward function, as well as in the effect of the anticipatory actions on the task execution efficiency of the human-robot team. For conditions 2 and 3 below, 15 subjects provided demonstrations by annotating human and robot actions.

6.1 Independent Variables

In our experiment, we controlled the robot decision-making mechanism in a human-robot collaborative task. This independent variable can have one of three values. which we refer to as "Manual session," "Auto1 session" and "Auto2 session."

1. Manual Control - The subject annotates robot actions by verbally commanding the robot.

2. Automatic Control with Manually Hand-coded Rewards (Auto1) - A domain expert observed the demonstrated sequences, manually specified the number of human types and explicitly specified a reward function that best explained the preference of each human type. A MOMDP policy was then computed, as described in Section 5.4. The robot automatically takes actions by executing that policy. The expert experimented several times by changing the reward function until the resulting policy would be satisfactory.

3. Automatic Control with Learned Rewards (Auto2) - We used clustering and inverse reinforcement learning to learn the reward function of the MOMDP and compute a policy, as described in sections 3, 4 and 5.

6.2 Hypotheses

H1 *Participants will agree more strongly that the robot takes anticipatory actions when working with the robot in the Auto1 and the Auto2 conditions, compared to working with the robot in the Manual condition.* The learned MOMDP policy enables the robot to infer online the preference of the human participant, and take anticipatory actions that match that preference. We expected the robot actions from the second and third condition to match the goals of the participants and to be strongly perceived as anticipatory, compared with manually jogging the robot. In prior work [5], MOMDPs have successfully been applied to recognizing human intention and using that information for decision-making.

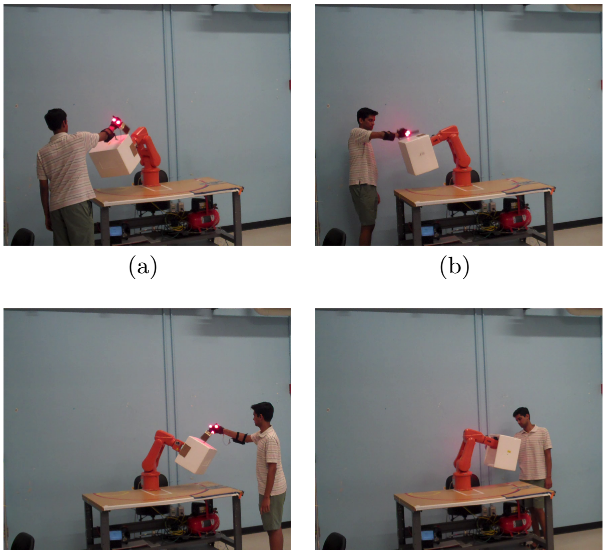
H2 *The robot performance as a teammate, as perceived by the participants, will be comparable between the Auto1 and Auto2 conditions.* We posited that the reward function learned by the proposed framework for each cluster of demonstrated action sequences would accurately represent the goals of participants with that preference, and would result in performance similar to that achieved using a reward function carefully annotated by a domain expert.

H3 *The team efficiency and fluency of human-robot teams of the Auto1 and Auto2 condition will be better than of the teams of the Manual condition.* We posited that the robot anticipatory actions of the learned MOMDP policy, as well as of the MOMDP policy from the hand-coded reward, would result in faster task execution and better team fluency compared with manually annotating robot actions. Automating robot behaviors [11] and, in particular, enabling anticipatory actions [12] has previously resulted in significant improvements in team efficiency and fluency in manufacturing applications.

6.3 Experiment Setting

We conducted a large-scale experiment of 36 human subjects to test the three hypotheses mentioned above, using a human-robot collaborative task. The human's role was to refinish the surface of a box attached to an ABB industrial robot. The robot's role was to position the box in a position that matches the human preference. All participants executed the task from four different positions, as shown in Figure 3. Each position required the participant to think of a different configuration for the box; therefore, the preference of the participant for the robot actions was dependent on his position. Our experiments used a within-subject design to mitigate the effects of inter-subject variability. We divided the subjects into balanced groups for each of the $k! = 6$ orderings of our $k = 3$ conditions. All experiment sessions were recorded. After each condition, all participants were asked to answer a post-session survey that used a five-point Likert scale to assess their responses to working with the robot. At the end of the experiment, participants also responded to an open-ended post-experimental questionnaire.

For the robot in the second and third conditions, the observable state variables x of the MOMDP framework were the box position along the horizontal and vertical axis, as well as the tilt angle of the box. The size of the observable state-space X was 726. The starting configuration of the box was specified by a small initial rotation and displacement from the center and level position. The demonstration data used to learn the MOMDP was provided prior to the experiment by 15 participants who were different than



(a)

(b)

(c)

(d)

Figure 3: Execution of a hand-finishing task by a human worker and an industrial robot. All participants were asked to execute the task from four different positions: (a) middle-left, (b) far-left, (c) middle-right and (d) far-right. In the first two positions (top), participants were asked to refinish the left surface of the box; in the other two (bottom), they were asked to refinish the right surface.

the 36 subjects. For the second condition, a domain expert grouped the participants into four types based on their final position. For the third condition, the clustering algorithm of the proposed framework identified four clusters based on whether the box was moved to the left or right side, as well as whether it was tilted and moved down or not. Therefore, including the partially observable variables, the total size of the state-space of the MOMDP was $4 \times 726 = 2904$ states. The robot actions corresponded to discrete changes in position and orientation of the box along three different axes. The human actions corresponded to the discretized motion of the human hand in the horizontal plane. To recognize the actions of the human, we used a Phasespace motion capture system consisting of eight cameras [25] that tracked the motion of a Phasespace glove worn by the participant. Measurements of the hand position were averaged in fixed intervals, and an action was detected when the difference between two consecutive averages exceeded a set threshold. Additionally, an oscillating hand-motion with a brush, simulating a surface refinishing, was recorded as the final action that ended the task. For both the second and third conditions, the observation function for the MOMDP was empirically specified. We leave learning an observation function from the demonstrated sequences for each cluster for future work.

7. RESULTS AND DISCUSSION

7.1 Subjective Measures

Two participants belonging to different groups did not answer the questions on the second page of their post-session surveys, and were therefore not taken into consideration. To maintain an equal number of participants per group, we randomly removed one participant from each of the remaining four groups, resulting in a final number of $n = 30$ participants.

Table 1: P-Values for three-way and pairwise comparisons (n=30). Statistically significant values are shown in bold.

Question	Omnibus	Auto2 v. Auto1	Auto2 v. Man	Auto1 v. Man
Q1	$\boldsymbol{p < 0.01}$	$\boldsymbol{p = 0.01}$	$p = 0.20$	$\boldsymbol{p = 0.01}$
Q2	$\boldsymbol{p < 0.01}$	$p = 0.34$	$\boldsymbol{p < 0.01}$	$\boldsymbol{p < 0.01}$

Q1: “The robot was responsive to me.”

Q2: “The robot anticipated my actions.”

As shown in Table 1, a three-way Friedman's test confirmed a statistically significant difference for question Q2. A pairwise Wilcoxon signed rank test with Bonferroni correction showed that, compared with manually annotating robot actions, participants agreed more strongly that the robot anticipated their actions when utilizing the proposed framework, Auto2 (Q2, $p < 0.01$), as well as when working with the robot in condition Auto1 (Q2, $p < 0.01$). These results follow from the fact that the MOMDP formulation allows the robot to reason over its uncertainty on the human type. Once the robot has enough information on the human type or preference, it will take actions toward task completion that follow that preference. Interestingly, the generated robot behavior emulates behavior frequently seen in human teams, where one member may let his teammate start working on a task, wait until he has enough information to confidently associate his teammates' actions with a familiar pattern based on prior experience, and then move onto task execution himself. This supports hypothesis **H1** of Section 6.2.

Recall our second hypothesis **H2**: that the robot performance as perceived by the participants would be comparable between conditions Auto1 and Auto2. We performed a TOST equivalence test, similarly to [10], and showed that participants rated robot intelligence ($p < 0.01$), accuracy of robot actions ($p = 0.02$), the robot's anticipation of participants' actions ($p = 0.03$), the trustworthiness of the robot ($p < 0.01$) and the smoothness of the robot's trajectories ($p = 0.04$) similarly between the two conditions. Interestingly, results from a pairwise Wilcoxon signed rank test indicated that participants agreed more strongly that the robot was responsive to them when working with the robot of the proposed framework (Auto2), compared with the robot that executed the MOMDP policy with the hand-coded reward function (Q1, $p = 0.01$). We believe that this is a result of the quality of the clustering algorithm: Whereas the domain expert grouped human subjects according to their final position upon completing the task, (Figure 3), our algorithm clustered subjects based on demonstrated human and robot action sequences. Therefore, when the robot of the proposed framework inferred the human preference and took anticipatory actions, it was perceived as more responsive.

7.2 Quantitative measures

Here, we consider the total task completion time and human idle time for each condition of the experiment. As Figure 4 shows, participants executed the task faster, on average, when working with the robot of the proposed framework (Auto2: $M = 102.8, SD = 8.33$), compared with the other two conditions (Auto1: $M = 111.4, SD = 8.85$ and Manual: $M = 107.4, SD = 33.0$). Furthermore, the human idle time was shorter when working with the robot of the proposed framework (Auto2: $M = 85.2, SD = 8.3$) compared with the other two conditions (Auto1: $M = 92.9, SD = 7.67$ and Manual: $M = 89.8, SD = 33.7$). (All units are in seconds.) A repeated-measure analysis of variance did not find the differences between the conditions in task execution and human idle time to be statistically significant. Additionally, no learning effects were observed as a function of session number.

Whereas this result does not provide adequate support for hypothesis **H3** of section 6.2, we observed that the variance of the task completion and human idle time between the subjects was much smaller when working with the robot of the proposed framework, compared to when manually annotating human and robot actions. We attribute this to the large variation in the amount of time subjects needed to find out which robot actions would bring the box to the desired position when manually controlling the robot. Additionally, some subjects annotated short trajectories during the manual condition, while others took multiple redundant steps when commanding the robot. Reducing the variation in execution time is important in the manufacturing domain, as it enables more efficient process planning and scheduling.

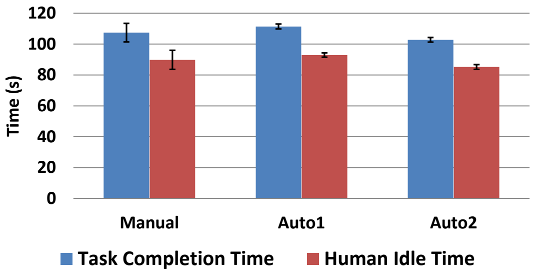


Figure 4: Average and standard error for task completion and human idle times in each condition.

We conducted a post-hoc experimental analysis of the data, and observed that task completion times varied significantly depending on the position from which participants were asked to execute the task (Figure 3). In particular, when manually annotating robot actions in the far-left position, a small number of robot motions within the horizontal plane were enough to bring the box from the starting point to a comfortable position. Indeed, a repeated measures analysis of variance with a Greenhouse-Geisser correction demonstrated statistically significant differences in task completion time between participants for the far-left position ($F(1.16, 46.0) = 26.61, p < 0.01$). Completion time at this position in the Manual condition was significantly lower than in the Auto2 condition ($t(29) = -6.651, p < 0.01$).

On the other hand, in the middle-right and far-right positions, several users spent a considerable amount of time during the Manual session trying to determine which robot actions would bring the box to a comfortable position from the starting configuration. A repeated measures analysis of variance with a Greenhouse-Geisser correction indicated statistically significant differences in task completion time as a function of condition at both the middle-left ($F(1.12, 32.5) = 7.03, p = 0.01$), and far-right positions ($F(1.66, 48.13) = 8.96, p < 0.01$). Completion time in the Auto2 condition was significantly lower than in the Manual condition at the middle-left ($t(29) = 3.16, p < 0.01$) and far-right ($t(29) = 3.39, p < 0.01$) positions (Figure 5). These results show that, at positions where the robot actions that would bring the box to a comfortable position were straightforward, simply annotating robot actions seemed to be the most efficient approach. For more complex configurations, the robot anticipatory behavior enabled by our framework resulted in a significant benefit to team efficiency.

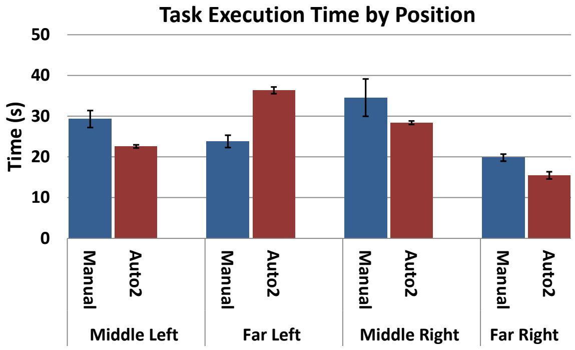


Figure 5: Average and standard error for the task completion times in Manual and Auto2 conditions.

7.3 Open-Ended Responses

When participants were asked to comment on their overall experience, some suggested that they appreciated the automated motion of the robot, because they found it difficult to determine how to move it to a favorable position. When asked to describe the part of the experiment that they liked most, some mentioned the ease of standing in place and allowing the robot to maneuver itself. One subject mentioned that, during the Manual session, she “had to think how to rotate the box, which turned out to be not trivial.” Several subjects, on the other hand, suggested that they preferred to be in control during the manual condition, rather than ceding control to the robot, and that both automated trials resulted in some unnecessary motion on the part of the robot. Interestingly, we observed an intermediate correlation between the execution time of the subjects in the manual condition and their preference for manually controlling the robot (r = -0.58). This result is indicative of a relationship between user preference and performance that warrants future investigation.

8. CONCLUSION

We presented a framework for automatically learning human user models from joint-action demonstrations, enabling the robot to compute a robust policy for a collaborative task with a human. First, we described the clustering of demonstrated action sequences into different human types using an unsupervised learning algorithm. These demonstrated sequences were used to learn a reward function that is representative for each type, through the employment of an inverse reinforcement learning algorithm. The learned models were then included as part of a MOMDP formulation, wherein the human type was a partially observable variable. In a human subject experiment ($n = 30$), participants agreed more strongly that the robot anticipated their actions when using the proposed framework ($p < 0.01$), compared with manually annotating robot actions. In trials where participants faced difficulty annotating the robot actions in order to complete the task, the proposed framework significantly improved team efficiency ($p < 0.01$). Also, compared with policies computed using a reward function hand-coded by a domain expert, the robot was more responsive to human actions when using our framework ($p < 0.01$). These results indicate that learning human user models through joint-action demonstrations and encoding them in a MOMDP formalism can support effective teaming in human-robot collaborative tasks.

While our assumption of full-observability of the task-steps is reasonable in a manufacturing setting with well-defined task procedures, many other domains including home-settings involve less structured tasks and would require extensions to our approach. Future work also includes the exploration of a hybrid approach between inferring "dominant" human types and learning an individualized user model: the robot starts with a policy corresponding to the general type of a new human user, and further refines its action selection mechanism through interaction.

9. REFERENCES

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

[2] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In *HRI*, 2012.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, May 2009.

[4] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *ICML*, pages 12–20, 1997.

[5] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus. Intention-aware motion planning. In *WAFR*. Springer, 2013.

[6] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Stud. Appl. Math. SIAM, June 1994.

[7] F. Broz, I. Nourbakhsh, and R. Simmons. Designing pomdp models of socially situated tasks. In *RO-MAN*, 2011.

[8] S. Chernova and M. Veloso. Teaching multi-robot coordination using demonstration of communication and state sharing. In *Proc. AAMAS*, 2008.

[9] F. Doshi and N. Roy. Efficient model learning for dialog management. In *Proc. HRI*, March 2007.

[10] A. Dragan, R. Holladay, and S. Srinivasa. An analysis of deceptive robot motion. In *RSS*, 2014.

[11] M. C. Gombolay, R. A. Gutierrez, G. F. Sturla, and J. A. Shah. Decision-making authority, team efficiency and human worker satisfaction in mixed human-robot teams. In *RSS*, 2014.

[12] G. Hoffman and C. Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *Proc. HRI*, 2007.

[13] V. Jääskinen, V. Parkkinen, L. Cheng, and J. Corander. Bayesian clustering of dna sequences using markov chains and a stochastic partition model. *Stat. Appl. Genet. Mol.*, 2013.

[14] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[15] B. Kim and J. Pineau. Maximum mean discrepancy imitation learning. In *Proceedings of RSS*, 2013.

[16] H. Kurniawati, D. Hsu, and W. S. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, pages 65–72, 2008.

[17] O. Macindoe, L. P. Kaelbling, and T. Lozano-Pérez. Pomcop: Belief space planning for sidekicks in cooperative games. In *AIIDE*, 2012.

[18] J. I. Marden. *Analyzing and modeling rank data*. CRC Press, 1995.

[19] T. B. Murphy and D. Martin. Mixtures of distance-based models for ranking data. *Computational statistics & data analysis*, 2003.

[20] T.-H. D. Nguyen, D. Hsu, W. S. Lee, T.-Y. Leong, L. P. Kaelbling, T. Lozano-Perez, and A. H. Grant. Capir: Collaborative action planning with intention recognition. In *AIIDE*, 2011.

[21] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proc. AAMAS*, 2003.

[22] S. Nikolaidis and J. Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proc. HRI*, 2013.

[23] S. C. Ong, Y. Grinberg, and J. Pineau. Mixed observability predictive state representations. In *AAAI*, 2013.

[24] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *IJRR*, 29(8):1053–1068, 2010.

[25] Phasespace, http://www.phasespace.com, 2012.

[26] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[27] G. Shani, J. Pineau, and R. Kaplow. A survey of point-based pomdp solvers. *In Proc. AAMAS*, 2013.

[28] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. In *Proc. NIPS*, 2007.

[29] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *J. Roy. Statist. Soc. Ser. B*, 2003.

[30] K. Waugh, B. D. Ziebart, and J. A. D. Bagnell. Computational rationalization: The inverse equilibrium problem. In *Proc. ICML*, June 2011.