Computational Formulation, Modeling and Evaluation of Human-Robot Team Training Techniques

by

Stefanos Z. Nikolaidis

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2014

Author .....
Department of Aeronautics and Astronautics
January 28, 2014

Certified by ....
Julie A. Shah
Assistant Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Computational Formulation, Modeling and Evaluation of Human-Robot Team Training Techniques

by

Stefanos Z. Nikolaidis

Submitted to the Department of Aeronautics and Astronautics
on January 28, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis is focused on designing mechanisms for programming robots and training people to perform human-robot collaborative tasks, drawing upon insights from practices widely used in human teams.

First, we design and evaluate human-robot cross-training, a strategy used and validated for effective human team training. Cross-training is an interactive planning method in which a human and a robot iteratively switch roles to learn a shared plan for a collaborative task. We present a computational formulation of the robot mental model, which encodes the sequence of robot actions towards task completion and the robot expectation over the preferred human actions, and show that it is quantitatively comparable to the human mental model that captures the interrole knowledge held by the human. Additionally, we propose a quantitative measure of human-robot mental model convergence, and an objective metric of mental model similarity. Based on this encoding, we formulate human-robot cross-training and evaluate it in human subject experiments ($n = 36$). We compare human-robot cross-training to standard reinforcement learning techniques, and show that cross-training provides statistically significant improvements in quantitative team performance measures. Additionally, significant differences emerge in the perceived robot performance and human trust. Finally, we discuss the objective measure of human-robot mental model convergence as a method to dynamically assess errors in human actions. This study supports the hypothesis that effective and fluent human-robot teaming may be best achieved by modeling effective practices for human teamwork.

We also investigate the robustness of the learned policies to randomness in human behavior. We show that the learned policies are not robust to changes in the human behavior after the training phase. For this reason, we introduce a new framework that enables a robot to learn a robust policy to perform a collaborative task with a human.The human preference is modeled as a hidden variable in a Mixed Observability Markov Decision Process, which is inferred from joint-action demonstrations of a collaborative task. The framework automatically learns a user model from training data, and uses this model to plan an execution policy that is robust to changes in the

human teammate's behavior. We compare the effectiveness of the proposed framework to previous techniques that plan in state-space, using data from the human subject experiments in which human and robot teams trained together to perform a place-and-drill task. Results demonstrate the robustness of the learned policy to increasing deviations in human behavior.

Thesis Supervisor: Julie A. Shah
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgments

Personal Acknowledgments

I would like to thank my advisor, Professor Julie Shah, for her fantastic supervision, guidance and support. Julie has been my role model of a young talented researcher, who has always reminded me that both intelligence and genuine interest for the students are necessary for a successful educator. Her constructive feedback has always been to the point and has defined the content of this thesis. Moreover, her kindness and enthusiasm have been an invisible force pushing me forward full-speed. It suffices to say that our Tuesday afternoon meeting has been one of my most cheerful times of the week! Looking back to when it all started, I am grateful for the amount of knowledge, technical and presentation skills and maturity that she helped me achieve. It has been truly a privilege.

Nothing of this would have been possible without the friendship and help of the Interactive Robotics Group in all aspects of my graduate life. Special thanks to Pem Lasota for being a great labmate and a companion in our gourmet marathons in France, to Dr. Jim Boerkoel for his essential feedback on research and karaoke nights in Tokyo, and to Ramya Ramakrishnan and Abhizna Butchibabu for the wonderful surprise party for my birthday!

I would also like to thank Professor Nick Roy, for giving me the opportunity to join the meetings of the Robust Robotics Group and share ideas with such a brilliant group of people! I am also thankful to Dr. Brad Knox for his essential feedback and our very exciting discussions.

Finally, I cannot thank enough my parents, Zachos and Efi, and my sister, Evelina, for their unconditioned love and support.

Funding

Funding for this work was partially provided by ABB. I would also like to acknowledge the Onassis Foundation as a sponsor.

Contents

List of Figures

Chapter 1

Introduction

Traditionally, industrial robots in manufacturing and assembly work in isolation from people. When this is not possible, the work is done manually. We envision a new class of manufacturing processes that achieve significant economic and ergonomic benefit through robotic assistance in manual processes. For example, mechanics in automotive and aircraft assembly spend a significant portion of their time retrieving and staging tools and parts for each job. A robotic assistant can provide productivity benefit by performing these non-value-added tasks for the worker. Other concepts for human and robot co-work envision large industrial robotic systems (examples in Fig. 1-1) that operate in the same physical space as human mechanics, as efficient and productive teammates.

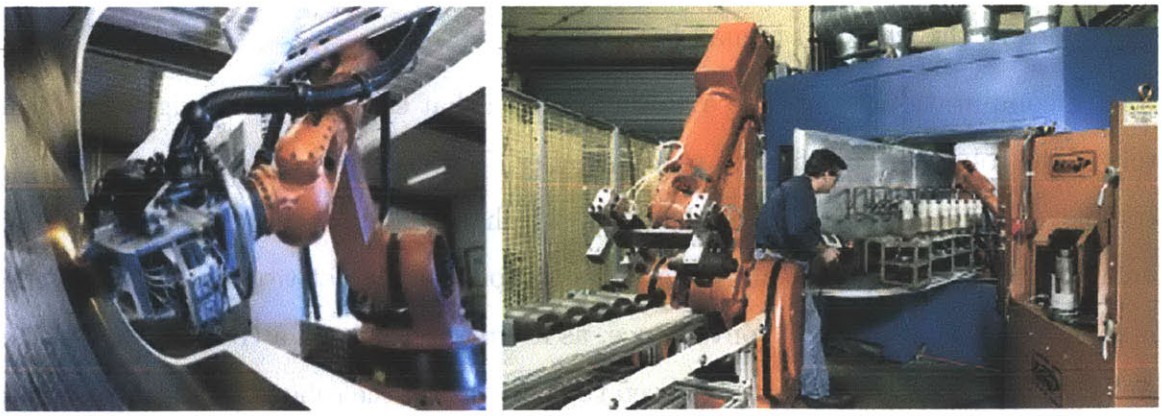

Figure 1-1: (Left) Coriolis Composite Placement Robot; (Right) Robotic Thermal Spraying of Parts

When humans work in teams, it is crucial for the members to develop fluent team behavior. We believe that the same holds for robot teammates, if they are to perform in a similarly fluent manner as members of a human-robot team. Learning from demonstration [5] is one technique for robot training that has received significant attention. In this approach, the human explicitly teaches the robot a skill or specific task [6, 1, 42, 15, 3]. However, the focus is on one-way skill transfer from a human to a robot, rather than a mutual adaptation process for learning fluency in joint-action. In many other works, the human interacts with the robot by providing high-level feedback or guidance [10, 26, 17, 54], but this kind of interaction does not resemble the teamwork processes naturally observed when human teams train together on interdependent tasks [34].

In this thesis, we focus on mechanisms inspired by human team training practices for programming robots and training people onto working together in collaborative tasks. First,we propose a training framework that leverages methods from human factors engineering, with the goal of achieving convergent team behavior during training and team fluency at task execution, as it is perceived by the human partner and is assessed by quantitative team performance metrics. Training with an actual robot on shared-location collaborative tasks could be dangerous or cost-prohibitive. Contrary to prior work, the training is done in a virtual environment, and we evaluate the team fluency when the human does the task with the actual robot after the training. Second, we expand the computational model, so that the robot can act robustly to deviations in the expected human behavior.

We computationally encode a teaming model that captures knowledge about the role of the robot and the human team member. The encoded model is quantitatively comparable to the human mental model, which represents the interrole knowledge held by the human [34]. Additionally, we propose quantitative measures to assess human-robot mental model convergence, as it emerges through a training process, as well as mental model similarity between the human and the robot. We then introduce a human-robot interactive planning method which emulates cross-training, a training strategy widely used in human teams [34]. We compare human-robot cross-training

to standard reinforcement learning algorithms through a large-scale experiment of 36 human subjects, and we show that cross-training improves quantitative measures of human-robot mental model convergence ($p = 0.04$) and mental model similarity ($p < 0.01$). Post-hoc experimental analysis shows that the proposed metric of mental model convergence could be used for dynamic human error detection. Additionally, a post-experimental survey shows statistically significant differences in perceived robot performance and trust in the robot ($p < 0.01$). Finally, we observe a significant improvement in team fluency metrics, including an increase of 71% in concurrent motion ($p = 0.02$) and a decrease of 41% in human idle time ($p = 0.04$), during the actual human-robot task execution phase that succeeds the human-robot interactive planning process. The improvement in team fluency is indicative of a transfer of the learning experience from a virtual environment to working with an actual robot.

In the next section, we discuss examples of human-robot interaction that motivate our work, and we place our work in context of other related work in Chapter 2. Chapter 3 presents our computational formulation of the human-robot teaming model, as well as methods to assess mental model convergence and similarity. Chapter 4 introduces human-robot interactive planning using cross-training, and describes the human subject experiments. In the same chapter we present and discuss the experiment results, which show a significant improvement in team performance using cross-training, as compared to standard reinforcement learning techniques. In Chapter 5, we exploit the fact that people generally have very few dominant preferences. Therefore, we formulate the interaction as a Mixed-Observability Markov Decision Process (MOMDP) framework, and show that planning in belief-space is more robust compared to planning in state-space. Finally, we conclude and present future work in Chapter 6.

Chapter 2

Human-Human and Human-Robot Team Training Practices

Our work is heavily inspired by human team training practices, applied prior to the execution of tasks or missions, with the goal of improving human team performance. We first present an overview of human team training techniques, and then review previous work on human-robot training.

2.1 Human Team Training Practices

In high-intensity domains, such as manufacturing, military and medical operations, there is a variety of tasks that are too complex or cognitively demanding to be performed by individuals working alone. To function as a team, individuals must coordinate their activities; simply bringing together several people to accomplish a task is not enough. Adaptive teams are able to coordinate their activities, not only under routine conditions, but also under novel conditions for which the teams have not been explicitly trained. Poor team coordination has been related to major system failures, such as in the cases of Three Mile Island and Chernobyl [16], where deficiencies in interaction and coordination resulted in failure to adapt to changes in the task environment. Studies of team training practices have mainly focused on improving team performance, particularly in response to novel event patterns.

One such technique is *procedural training*: A form of process training in which "operators in complex systems are positively reinforced (through feedback) to follow a standard sequence of actions each time a particular stimulus is encountered." [20] Trainees practice by repetitively following prespecified procedures, with the goal of learning to respond automatically to stimuli. The underlying assumption is that training in this manner reduces the incidence of errors and enhances performance [22]. Procedural training is prevalent in medical, manufacturing and military settings, for tasks in which deviations from complicated procedures can be catastrophic. Whereas this type of training enables team members to reflexively react under stressful conditions and a heavy workload, it is argued that it can also limit a team's ability to transfer training to novel situations, leading to poor performance when the actual task execution conditions do not match the training conditions [20].

In *cross-training*, another common technique, team members are trained for each other's roles and responsibilities, in addition to their own [9]. There are three types of cross-training: (a) positional clarification, (b) positional modeling, and (c) positional rotation. Positional clarification involves verbally presenting team members with information about their teammates jobs through lecture or discussion. Positional modeling includes observations of team-members' roles through videotape or direct observation. Positional rotation is the most in-depth form of cross-training. Findings [34, 13] suggest that positional rotation cross-training, defined as "learning interpositional information by switching work roles," is strongly correlated with improvement in human team performance, as it provides individuals with hands-on knowledge about the roles and responsibilities of their teammates [34]. Positional rotation cross-training has been used by military tactical teams, as well as aviation crews. It has been argued that shared expectations, resulting from the development of shared knowledge, allow team members to generate predictions for appropriate behavior under novel conditions and in cases when there is uncertainty in the information flow [34]. The proposed human-robot cross-training algorithm is inspired by the positional-rotation type of training practice. Whereas in this work we do not examine novel situations in tasks performed by human-robot teams, uncertainty is

present due to the inherent lack of transparency between human and robot. Additionally, task execution following training is conducted in an actual environment, which is inherently different to the virtual environment where training takes place. From an algorithmic point of view, switching roles has the additional benefit of enabling the human to directly demonstrate his preference, as explained in Section 4.4.2.

While cross-training is feasible for small teams, it can become impractical as teams grow in size. Recently, Gorman et al. introduced *perturbation training* [20]. Using this approach, standard coordination procedures are disrupted multiple times during the training process, forcing team members to coordinate in novel ways to achieve their objective. Perturbation training aims to counteract habituation associated with task processes a possible outcome of procedural training. It is inspired by prior work in motor and verbal learning, and is aimed at improving team performance under novel post-training conditions [48]. A form of perturbation training was examined in [21], in which teams were trained to perform a repetitive command-and-control task during two training rounds. Teams that changed members for the second round developed higher quality team processes after training, compared to teams that maintained the same members during both rounds. Gorman et al. used a different form of perturbation for an air reconnaissance task, by temporarily limiting communication between team members and disabling available equipment [20]. The performance of teams that received perturbation-training was compared to those that received procedural or cross-training. Whereas teams that cross-trained exhibited an increase in interpositional teamwork knowledge across sessions and performed better in one of three missions than those that underwent procedural training, those that received perturbation training exhibited better performance in two of three missions. The Gorman study suggests that perturbation training is a very promising training method, although a larger variety of experiments may be required to obtain conclusive evidence. We believe that it would be interesting to introduce perturbation training in a human-robot team setting, but leave the notion for future investigation.

2.2 Human-Robot Team Training

While there has been extensive work conducted on human team training techniques, in human-robot team settings training has focused on one-way knowledge given by a human teacher to a robot apprentice. An example of this method is the SARSA(λ) reinforcement learning approach, where the reward signal is interactively assigned by the human. This technique falls into the category of learning wherein the **human and machine engage in high-level evaluation and feedback**.

In some approaches within this category, a human trainer assigns positive reinforcement signals [10] to a virtual character - a method also known as "clicker training." The state space is represented by a percept tree, which maintains a hierarchical representation of sensory input. The leaf nodes represent the highest degree of specialization, and the root node matches any sensory input. Similarly, state-action pairs consisting of percepts that generate the same action are organized hierarchically, according to the specificity of the percept. Each state-action pair is assigned a reward depending on whether it has good, bad or indifferent consequences. The structure of the percept tree and the rewards is refined interactively by a human trainer. A similar approach is detailed in [26], wherein clicker training is used to train four-legged robots. In this proposed system, the behavior of the robot is implemented through a hierarchical tree of schemata, where each schema is constituted by a set of activation conditions and a set of executable actions. Human feedback is then used to create new behaviors through the combination of existing ones. The robot maintains a user-specific model of human behavior that is updated through interaction and affects the probability of transitions between different schemata. A user model is also learned in [17], simultaneously with a dialog manager policy in a robotic wheelchair application. The model is encoded in the transition and observation functions and rewards of a Partially Observable Markov Decision Process framework. The hidden state represents the user's intent; that is, the places where the user would like the wheelchair to go. The human interacts with the system by giving verbal commands, as well as a scalar reward after each robot action.

Other methods, such as TAMER-RL [28, 29], support the use of human input to guide a traditional reinforcement learning agent in maximizing an environmental reward. The TAMER framework is based upon two insights into how humans assign rewards: First, human reward is delayed according to the time it takes the trainer to evaluate behavior and deliver feedback. Second, a human assigns rewards after considering their long-term effects; in that sense, the reward value more resembles a state-action value than an environmental reward in the manner of a Markov Decision Process (MDP) framework [51]. SARSA(λ) is augmented by different approaches of combining human reward in TAMER-RL, and their effectiveness is tested through experiments involving a mountain-car and cart-pole task. Q-Learning with Interactive Rewards [54] is identical to our version of SARSA(λ), if we remove eligibility traces on SARSA and set a greedy policy for both algorithms. In this case, the algorithm has been applied to teach a virtual agent to cook from a recipe, with the human assigning rewards to the agent by moving the green slider on a vertical bar. A modified version [54] incorporating human guidance has been empirically shown to significantly improve several dimensions of learning. That version of the algorithm resulted in fewer failures, as the learning process was focused on smaller, more relevant parts of the state-space.

The other category for learning in human-robot teams involves a **human providing demonstrations to the machine**. Work involving learning from demonstration includes systems that learn a general policy for a task by passively observing a human expert executing that task. For example, in [14] the system learns a Gaussian Mixture Model for each action class, using human demonstrations as training data. Each new datapoint is assigned to a mixture class according to maximum likelihood. The algorithm also returns a confidence measure, used by the agent to request additional demonstrations. This proposed algorithm is improved through automatic selection of multiple confidence thresholds in [15]. More recently, Gaussian Mixture Models [3] have been used to teach a skill to a robot during experiments in which a human physically guides the robot through a trajectory; this approach is known as "kinesthetic teaching." These experiments have shown that guiding a robot arm

through keyframes is a more effective method of teaching means-oriented skills, such as performing gestures, than guiding the robot through the entire trajectory. This is partially due to the difficulty in smoothly manipulating a heavy robot arm. However, demonstrating an entire trajectory has been more successful for goal-oriented skills, such as pick-and-place tasks.

Another approach to robot training is to teach a plan to the robot. In [42], the authors assume that the robot has an available set of low-level behaviors. Given this assumption, the goal is then for the robot to build a high-level task representation of a more complex, sequentially structured task using its existing behavior set. The robot learns the necessary tasks by creating a link between observations and robot behaviors that achieve the observed effects. Used in addition to human demonstrations, instructional feedback focuses the learning process on the relevant aspects of a demonstration. In [42], experiments in which a Pioneer 2-DX mobile robot attempts to complete a pick-and-place task validate the correctness of learned representations.

In another training method, the robot learns a system model that consists of a transition model from state s given action a, $T(s'/s, a)$, and a reward function $R(s)$ which maps states to a scalar reward. Using this system model, a policy that maps states to actions can maximize the finite- or infinite-horizon accumulated reward. Atkeson and Schaal consider the problem of having a robot arm follow a demonstrated trajectory [6]. In their paper, the robot learns the transition model through repeated attempts to execute the task, and the reward function is modeled so as to quadratically penalize deviation from the desired trajectory. A priori human knowledge was used to divide a vertical balancing task into a swing-up component and a balancing component. Results from experiments indicate improved performance compared to simply mimicking demonstrated motions.

Apprenticeship learning [1] generalizes to task planning applications, employing a Markov decision process framework. In this method, the algorithm assumes that the expert tries to maximize a "true" unknown reward function that can be expressed as a linear combination of known "features". A quadratic program is solved iteratively to find feature weights that attempt to match the expected feature counts of the

resulting policy with those of the expert demonstrations. Experiment results using this approach indicate that robot performance is similar to that of the expert, even though the expert reward function may not be recovered. This work falls into the category of inverse reinforcement learning (IRL), wherein the MDP state reward function is derived from observed expert demonstrations [40]. In multi-agent settings, state-of-the-art behavior modeling based on the game-theoretic notion of regret and the principle of maximum entropy has accurately predicted future behavior in newly encountered domains [56].

Our proposed human-robot cross-training algorithm uses the same inputs as Q-learning with Interactive Rewards [54]. However, rather than asking the human to explicitly provide feedback to the robot, the human feedback is instead provided implicitly by having the human switch roles with the robot, in a manner similar to effective human team-training practices. This part of training resembles Inverse Reinforcement Learning [40], as the state reward function is learned by human demonstrations when the human and robot switch roles. A key difference from previous work, however, is that we focus on collaborative tasks wherein robot and human actions are interdependent. Therefore, the outcome of the robot actions depends on the human actions, and leads to a learned model of human actions encoded in the Transition Probabilities of a Markov Decision Process Framework similar to that observed in [17]. By following a human-team inspired approach, we support the mutual co-adaptation of both the human and robot, and focus on the team-fluency in shared-location, joint-action collaborative tasks, rather than the optimization of agent performance metrics.

Chapter 3

Mental Model Formulation

In this chapter, we describe the concept of shared mental models in human teams, and then computationally encode a mental model for the robot as a Markov Decision Process. Based on this encoding, we then introduce an objective measure - the entropy rate of the Markov chain - to evaluate the convergence of the human and robot mental models. Finally, we propose a metric for human-robot mental model similarity inspired by shared mental model elicitation techniques in human teams.

3.1 Shared Mental Models in Human Teams

The literature presents various definitions for the concept of shared mental models [31]. Marks et al. [34] state that mental models represent "the content and organization of interrole knowledge held by team members within a performance setting." According to [37], mental models are "mechanisms whereby humans generate descriptions of system purpose and form, explanations of system functioning and observed system states, and prediction of future system states, ... and they help people to describe, explain, and predict events in their environment." The objective of team training is to foster similar or shared mental models, as empirical evidence suggests that mental model similarity improves coordination processes, which, in turn, enhances team performance. Most researchers agree that there are multiple types of mental models shared among team members. [37] state that one type is technol-

ogy/equipment mental models that capture the dynamics and control of the technology among team members. Task mental models describe and organize knowledge about how a task is accomplished in terms of procedures and task strategies, whereas team interaction models describe the roles and responsibilities of team members. Finally, team mental models capture team-specific knowledge of teammates, such as their skills and preferences. In this work, we refer to robot mental model as the learned sequence of robot actions toward task completion, as well as the expectation that the robot has for human actions. We computationally encode this model as a Markov Decision Process (MDP) [47].

3.2 Robot Mental Model Formulated as MDP

We describe how a robot teaming model can be computationally encoded as a Markov Decision Process. A Markov decision process is a tuple $\{S, A, T, R\}$, wherein:

- S is a finite set of world states; it models the set of world environment configurations.
- A is a finite set of actions; this is the set of actions the robot can execute.
- $T : S \times A \longrightarrow \Pi(S)$ is the state transition function, giving a probability distribution over world states for each world state and action; the state transition function models the uncertainty that the robot has in the human action. For a given robot action a, the human's next choice of action yields a stochastic transition from state s to a state s'. We write the probability of this transition as $T(s, a, s')$. In this formulation, human behavior is the cause of randomness in our model, although this can be extended to include stochasticity from the environment or the robot actions.
- $R : S \times A \longrightarrow \mathbb{R}$ is the reward function, giving the expected immediate reward gained by performing each action in each state. We write $R(s, a)$ for the expected reward for taking action a in state s.

The policy π of the robot is the assignment of an action $\pi(s)$ at every state s. The optimal policy π^* can be calculated using dynamic programming [47]. Under this formulation, the role of the robot is represented by the optimal policy π^*, whereas robot knowledge of the role of the human co-worker is represented by the transition probabilities T.

3.3 Evaluation of Mental Model Convergence

As the mental models of the human and robot converge, we expect the human and robot to perform similar patterns of actions. This means that the same states will be visited frequently and robot uncertainty about human action selection will decrease. Additionally, if the mental models of the human and the robot converge, the patterns of actions performed will match the human preference, as elicited after the training.

To evaluate the convergence of the robot's computational teaming model and the human mental model, we assume a uniform prior and compute the *entropy rate* [18] of the Markov chain (Eq. 3.1). The Markov chain is induced by specifying a policy π in the MDP framework. For the policy π, we use the robot actions that match human preference as elicited by the human after training with the robot. Additionally, we use the states $s \in S$ that match the preferred sequence of configurations for task completion. For a finite state Markov chain X with initial state s_0 and transition probability matrix T, the entropy rate is always well-defined [18]. It is equal to the sum of the entropies of the transition probabilities $T(s, \pi(s), s')$, for all $s \in S$, weighted by the probability of the occurrence of each state according to the stationary distribution μ of the chain (Equation 3.1).

$$H(X) = -\sum_{s \in S} \mu(s) \sum_{s' \in S} T(s, \pi(s), s') log\left[T(s, \pi(s), s')\right] \tag{3.1}$$

Interestingly, the conditional entropy given by Eq. 3.1 also represents the uncertainty of the robot about the action selection of the human, which we expect to decrease as human and robot train together. This measure can be generalized to

encode situations in which the human has multiple preferences or acts stochastically. In Section 4.4.1, we conduct a post-hoc analysis indicating entropy evolution over time in such cases. The entropy rate appears to be particularly sensitive to changes in human strategy, and reflects the resulting increase in robot uncertainty about the next actions of the human. We propose that these results provide intriguing first support for the potential use of entropy rate as a component of a human error detection mechanism.

3.4 Human-Robot Mental Model Similarity

Given the formulation of the robot mental model, we propose a similarity metric between human and robot mental models based on prior work [31] on shared mental model elicitation for human teams. In a military simulation study [35], each participant was asked to annotate a sequence of actions that he and his teammates should follow to achieve mission completion. The degree of mental model similarity was then calculated by assessing the overlap in action sequences selected by each of the team members. We generalize this approach in a human-robot team setting: In our study, the participant annotates a sequence of actions that he or she thinks the human and robot should perform in order to complete the assigned task. We then elicit the similarity of the human and robot mental models by determining the ratio of annotated robot actions matching the actions assigned by optimal policy to the total number of robot actions required for task completion. This describes how well human preference for robot actions matches the actual optimal policy for the MDP.

Chapter 4

Human-Robot Cross-Training

In the computational encoding of the mental model for the robot described in the previous chapter, expert knowledge about task execution is encoded in the assignment of rewards R, and in the priors on the transition probabilities T that encode the expected human behavior. This knowledge can be derived from task specifications or from observation of expert human teams. However, rewards and transition probabilities finely tuned to one human worker are not likely to generalize to another human worker, since each worker develops his or her own highly individualized method for performing manual tasks. In other words, a robot that works with one person according to another person's preferences is not likely to be good teammate. Empirical evidence suggests that mental model similarity improves coordination processes, which in turn enhance team performance [34]. Mental model similarity is particularly important under conditions in which communication is difficult due to excessive workload, time pressure or another environmental feature, as teams are unable to engage in necessary strategizing in these circumstances [36]. Shared or similar mental models are important in such cases, as they allow team members to predict the information and resource requirements of their teammates. In the case of a human-robot team, communication is difficult for different reasons: Transparency in the interaction between human and robot is an unsolved problem, mainly due to the technical challenges inherent in the exchange of information about high-level goals and intentions between human and robot. We therefore hypothesize that a shared-

mental model for a human-robot team will improve team performance in actual task execution. Cross-training [34] is a validated and widely used mechanism for conveying shared mental models in human teams; we emulate the cross-training process that takes place among human team-members by having the human and robot train together in a virtual environment. We use a virtual environment because, especially in high-intensity applications, it is infeasible or cost-prohibitive for a robot to perform the human's role in an actual environment, and vice versa.

4.1 Cross-Training Emulation in Human-Robot Team

We emulate positional rotation in human teams by having the human and robot iteratively switch roles. We name the phase in which the human and robot roles match those of the actual task execution as the *forward phase*, and the phase in which the human and robot roles are switched as the *rotation phase*. In order for the computational teaming model of the robot to converge with the human mental model:

1. The robot must have an accurate estimate of the role of the human in performing the task. We use the human-robot forward phase of the training process to update our estimation of the transition probabilities that encode the expected human behavior.

2. The actions of the robot must match the human preference. We accomplish this by including human inputs in the rotation phase to update the reward assignments.

4.1.1 Human-Robot Cross-Training Algorithm

The Human-Robot Cross-training algorithm is summarized in Figure 4-1. In Line 1, rewards $R(s, a)$ and transition probabilities $T(s, a, s')$ are initialized from prior knowledge about the task. In Line 2, an initial policy π is calculated for the robot; we used value iteration [47] in our implementation. In Line 4, the Forward-phase

Algorithm : Human-Robot Cross-training

1. Initialize $R(s, a)$ and $T(s, a, s')$ from prior knowledge
2. Calculate initial policy π
3. **while**(number of iterations < *MAX*)
4. Call Forward-phase(π)
5. Update $T(s, a, s')$ from observed sequence $s_1, a_1, s_2, \ldots, s_{M-1}, a_{M-1}, s_M$
6. Call Rotation-phase()
7. Update $R(s_i, a_i)$ for observed sequence $s_1, a_1, s_2, a_2, \ldots, s_N, a_N$
8. Calculate new policy π
9. **end while**

Figure 4-1: Human-Robot Cross-Training Algorithm

function is called, where the human and robot train for the task. The robot chooses its actions depending on the current policy π, and the observed state-action sequence is recorded. In Line 5, $T(s, a, s')$ are updated based on the observed state-action sequence. $T(s, a, s')$ describes the probability that, for a task configuration modeled by state s and robot action a, the human will perform an action such that the next state will be s'.

In the rotation phase (Line 6), the human and robot switch task roles. In this phase, the observed actions $a \in A$ are the actions performed by the human worker, whereas the states $s \in S$ remain the same. In Line 7, the rewards $R(s, a)$ are updated for each observed state s and human action a. We then use the new estimates for $R(s, a)$ and $T(s, a, s')$ to update the current policy (Line 8). The new optimal policy is computed using standard dynamic programming techniques [47].

In our implementation, we update the rewards (Line 7) as follows:

$$R(s, a) = R(s, a) + r \tag{4.1}$$

The value of the constant r needs to be large enough, compared to the initial values of $R(s, a)$, for the humans actions to affect the robot's policy. Note that our goal is not to examine the best way to update the rewards, as this has proven to be task-dependent [29]. Instead, we aim to provide a general human-robot training framework, and use

the reward update of Eq. 4.1 as an example. Knox and Stone [28] evaluate eight methods for combining human inputs with MDP reward in a reinforcement learning framework. Alternatively, inverse reinforcement learning algorithms could be used to estimate the MDP rewards from human input [1].

We iterate the forward and rotation phases for a fixed number of MAX iterations, or until a convergence criterion is met.

4.1.2 Forward Phase

The pseudocode of the forward phase is presented in Figure 4-2. In Line 1, the current state is initialized to the start step of the task episode. The FINAL_STATE in Line 2 is the terminal state of the task episode. In Line 3, the robot executes an action a assigned to a state s, based on the current policy π. The human action is observed (Line 4) and the *next_state* variable is set according to the *current_state*, the robot action a and the human action. In our implementation, we use a look-up table that sets the next state for each state and action combination. Alternatively, the next state could be directly observed after the human and robot finish executing their actions. The state, action, and next state of the current time-step are recorded (Line 6).

4.1.3 Rotation Phase

The pseudocode of the rotation phase is presented in Figure 4-3. In Line 3, the action a is the observed human action. In Line 4, a robot action is sampled from the transition probability distribution $T(s, a, s')$.

Just as the transition probability distributions of the MDP are updated after the forward phase, the robot policy is updated to match the humans expectations after the rotation phase. This process emulates how a human mental model would change while working with a partner. A key feature of the cross-training approach is that it also provides an opportunity for the human to adapt to the behavior of the robot.

Function: Forward-phase(policy π)
1. Set *current_state* = START_STATE
2. **while**(*current_state* != FINAL_STATE)
3. Execute robot action *a* according to current policy π
4. Observe human action
5. Set *next_state* to the state resulting from *current_state*, robot and human action
6. Record *current_state*, *a*, *next_state*
7. *current_state* = *next_state*
8. **end while**

Figure 4-2: *Forward Phase* of the Cross-Training Algorithm

4.1.4 Reinforcement Learning with Human Reward Assignment

We compare the proposed formulation to the interactive reinforcement learning approach, wherein the reward signal of an agent is determined by interaction with a human teacher [55]. We use SARSA(λ) with greedy policy [51] as the reinforcement learning algorithm, due to its popularity and applicability to a wide variety of tasks. In particular, SARSA(λ) has been used to benchmark TAMER framework [27], as well as to test TAMER-RL [28, 29]. Variations of SARSA have been used to teach a mobile robot to deliver objects [46], for navigation of a humanoid robot [39] and in an interactive learning framework, wherein the user gives rewards to the robot through verbal commands [53]. Furthermore, our implementation of SARSA(λ) would be identical to Q-Learning with Interactive Rewards [54], if we removed eligibility traces on SARSA and, in the case of a greedy policy, for both algorithms.

After each robot action, the human is asked to assign a good, neutral, or bad reward $\{+r, 0, -r\}$. In our current implementation, we set the value of r, the reward signal assigned by the human, to be identical to the value of the reward update in cross-training (Eq. 4.1 in Section 4.1) for comparison purposes.

```
Function: Rotation-phase()
1.  Set current_state = START_STATE
2.  while(current_state != FINAL_STATE)
3.          Set action a to observed human action
4.          Sample robot action from T(current_state, a, next_state)
5.          Record current_state, a
6.          current_state = next_state
7.  end while
```

Figure 4-3: *Rotation Phase* of the Cross-Training Algorithm.

4.2 Human-Robot Teaming Experiments

We conducted a large-scale experiment ($n = 36$) to compare human-robot cross-training to standard reinforcement learning techniques.

4.2.1 Experiment Hypotheses

The experiment tested the following four hypotheses about human-robot team performance:

- Hypothesis 1: Human-robot interactive planning with cross-training will improve quantitative measures of **human-robot mental model convergence** and **mental model similarity** compared to human-robot interactive planning using reinforcement learning with human reward assignment. We base this hypothesis on prior work indicating that cross-training improves similarity of mental models among human team members [34, 13].

- Hypothesis 2: Participants who cross-trained with the robot will agree more strongly that **the robot acted according to their preferences**, compared to participants who trained with the robot by assigning rewards. Furthermore, we hypothesize that they will agree more strongly that **the robot is trustworthy**. We base this hypothesis upon prior work [49] that indicated that humans

find the robot more trustworthy when it emulates the effective coordination behaviors observed in human teams.

- Hypothesis 3: Human-robot interactive planning with cross-training will improve **team-fluency metrics on task-execution** compared to human-robot interactive planning using reinforcement learning with human reward assignment. We base this hypothesis on the wide usage of cross-training to improve performance in human teams [34].

- Hypothesis 4: The **learning experience within a virtual environment** of training with a robot will **transfer** to an improvement in team fluency metrics and subjective performance measures, when **working with the actual robot** at the task execution phase.

4.2.2 Experiment Setting

As a proof of concept, we applied the proposed framework to train a team of one human and one robot to perform a simple place-and-drill task. In the task, there were three positions that could either remain empty or have a screw placed or drilled into them. The human's potential actions included either the placement of a screw in one of the empty holes, or waiting (no-action), while the robot could either drill each placed screw or wait.

Although this task is simple, we found it adequate for the testing of our framework as there is a sufficient variety of ways to accomplish the task among different persons. For example, some participants preferred to place all screws in sequence from right-to-left and then have them drilled in the same sequence, while others preferred to place and drill each screw before moving on to the next. For the humans who preferred to place all three screws first before drilling, there are $3! \times 3! = 36$ different potential orderings for the placement and drilling of the screws. For those who annotated as their preference to have a screw drilled immediately after placement, there are $3! = 6$ different possible orderings. Therefore, there are a total of $36 + 6 = 42$ different potential orderings for these two high-level strategies. This is a lower bound

on the possible human preferences for this task, as it does not include the case of a mixed strategy, where the human preferred to have the robot drill one screw immediately after placement, but only drill the remaining screws after they had all been placed. The participants consisted of 36 subjects recruited from MIT. Videos of the experiment can be found at: **http://tiny.cc/5q685w**

4.2.3 Human-Robot Interactive Training

Before initiating training, all participants were asked to describe, both verbally and in written form, their preferred method of executing the task. We then initialized the robot policy using a set of prespecified policies, in a way clearly different from the participant's preference. We did this to avoid the potential trivial case in which the initial policy of the robot matches the preferred policy of the user, and also to evaluate mental model convergence starting from different human and robot mental models.

The participants were randomly assigned to two groups: Group A and Group B. Each participant then underwent a training session within the ABB RobotStudio virtual environment, where the human controlled the white anthropomorphic "Frida" robot depicted on the left in (Figure 4-4) while working with the orange industrial robot, "Abbie," on the right. The human chose an action in discrete time steps and observed the outcome by watching "Frida" move concurrently with "Abbie." The motions of both human and robot actions were predefined, and there was a single motion for each action.

Depending on the assigned group, the participant underwent one of the following training sessions:

1. Cross-training session (Group A): The participant iteratively switched positions with the virtual robot, placing the screws during the forward phase and drilling during the rotation phase.

2. Reinforcement learning with human reward assignment session (Group B): This is the standard reinforcement learning approach, wherein the participant placed

screws and the robot drilled at all iterations, with the participant assigning a positive, zero, or negative reward after each robot action [17].

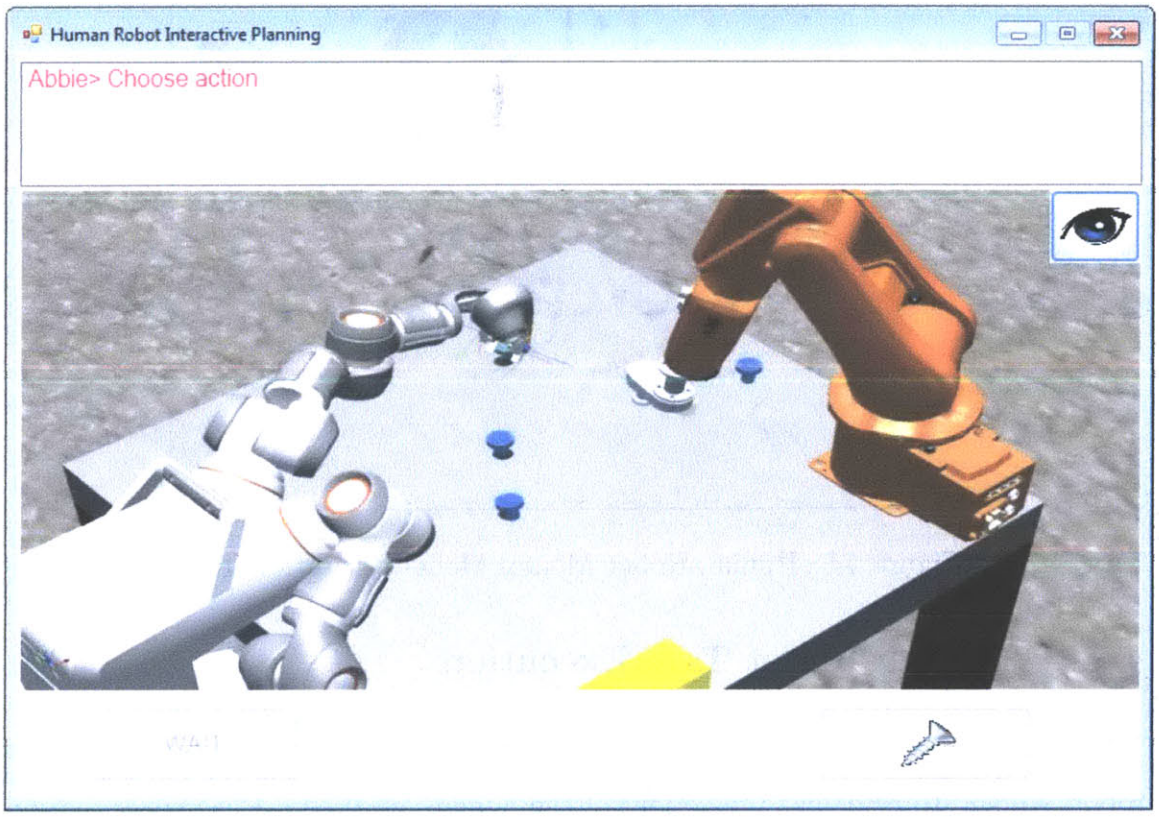

Figure 4-4: Human-Robot Interactive Planning Using ABB RobotStudio Virtual Environment. The human participant controls the white anthropomorphic "Frida" robot on the left, to work with the orange industrial robot, "Abbie," on the right.

For the cross-training session, the policy update (Line 8 of Figure 4-1, Section **??**) was performed using value iteration with a discount factor of 0.9. The SARSA(λ) parameters in the standard notation of SARSA [51] were empirically tuned ($\lambda = 0.9, \gamma = 0.9, \alpha = 0.3$) for optimal task performance.

After the training session, the mental model of all participants was assessed using the method described in Section 3.4. For each workbench configuration through task completion, participants were asked to choose a placing action and their preference for an accompanying robot drilling action, based on the training they had experienced together (Figure 4-5).

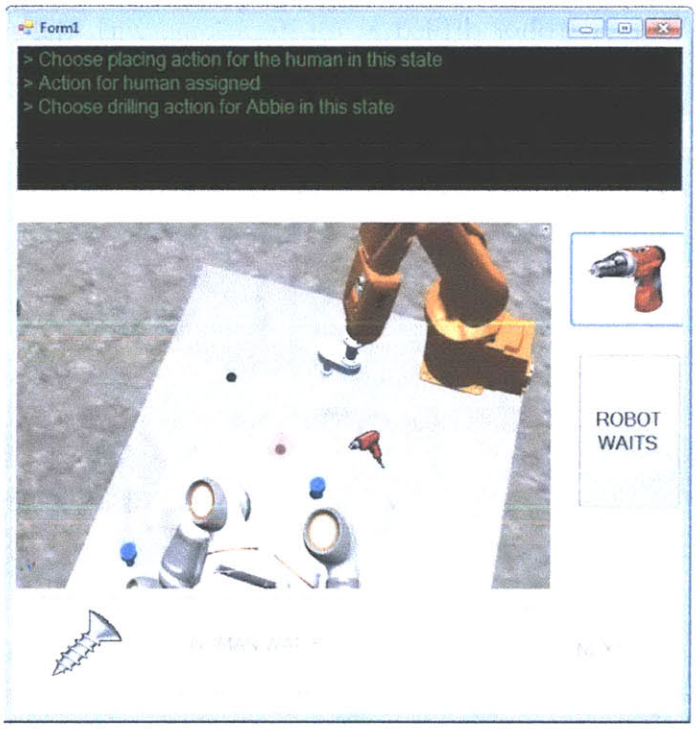

Figure 4-5: Human-Robot Mental Model Elicitation Tool

4.2.4 Human-Robot Task Execution

We then asked all participants to perform the place-and-drill task with the actual robot, Abbie. To recognize the actions of the human, we used a Phasespace motion-capture system of eight cameras [45] that tracked the motion of a Phasespace glove worn by the participant (Figure 4-6). Abbie executed the policy as learned from the training sessions. The task execution was videotaped and later analyzed for team fluency metrics. Finally, all participants were asked to respond to a post-experiment survey.

4.3 Results and Discussion

Results of the human subject experiments indicate that the proposed cross-training method outperforms standard reinforcement learning in a variety of quantitative and qualitative measures. This is the first evidence that human-robot teamwork is improved when a human and robot train together by switching roles in a manner similar

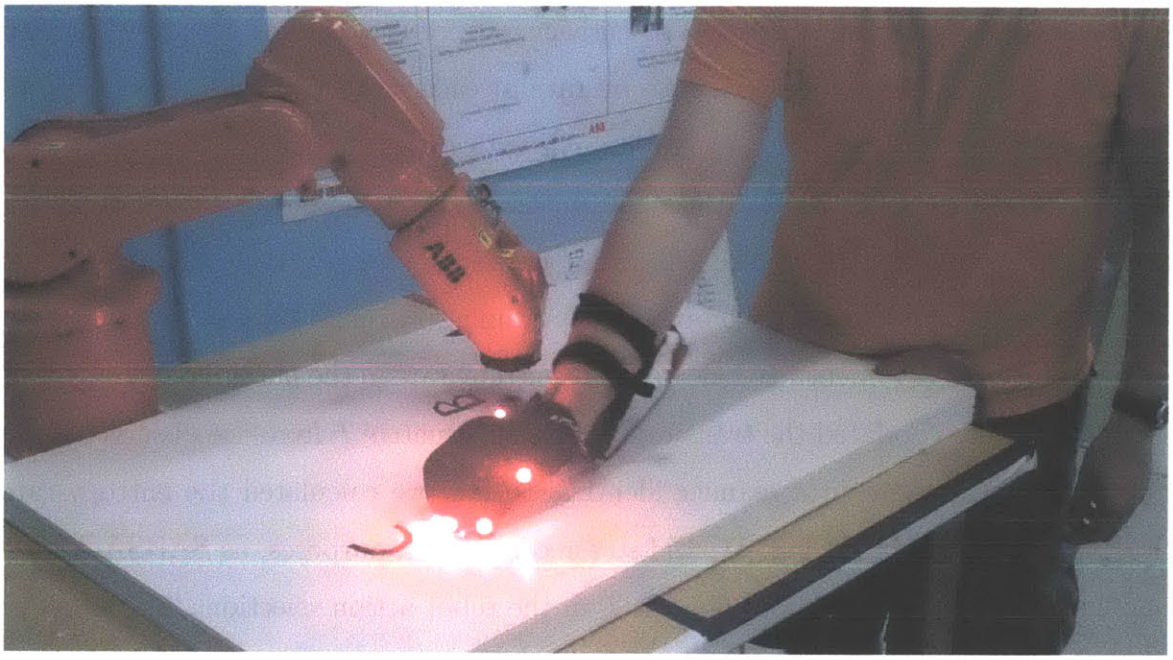

Figure 4-6: Human-Robot Task Execution

to effective human team training practices. Unless stated otherwise, all the p-values in this section are computed for *two-tailed, unpaired t-tests with unequal variance.*

4.3.1 Quantitative Measures

Mental Model Similarity

As described in Section 3.4, we computed the mental model similarity metric as the ratio of human drilling actions matching the actions assigned by the robot policy to the total number of drilling actions required for task completion. Participants in Group A had an average ratio of 0.96, compared to an average ratio of 0.75 in Group B ($p < 0.01$). This shows that participants who cross-trained with the robot developed mental models more similar to the robot teaming model than participants who trained with the robot by assigning rewards.

Mental Model Convergence

Mental model similarity was also reflected by similar patterns of actions observed during the training process, and by decreased robot uncertainty about the human's

action selection, as computed by the entropy rate of the Markov Decision Process (Section 3.3). We computed the entropy rate at each training round using the preferred robot policy, as elicited by the human with the mental model elicitation tool (Figure 4-5 of Section 4.2.3). Since the initial value of the entropy rate varies for different robot policies, we used the mean percent decrease across all participants of each group as a metric to compare cross-training to reinforcement learning with human reward assignment. To calculate the entropy rate in the human reward assignment session, we updated the transition probability matrix T from the observed state and action sequences, in a manner identical to how we calculated the entropy-rate for the cross-training session. We did so for comparison purposes, as SARSA(λ) is a model-free algorithm and does not use T in the robot action selection [51].

Figure 4-7 shows the entropy rate after each training round for participants in both groups. We considered only the 28 participants who did not change their preference. The difference between the two groups after the last training round is statistically significant ($p = 0.04$), indicating that the robot's uncertainty about the human participant's actions after training was significantly lower for the cross-training group than in the group that used reinforcement learning with human reward assignment.

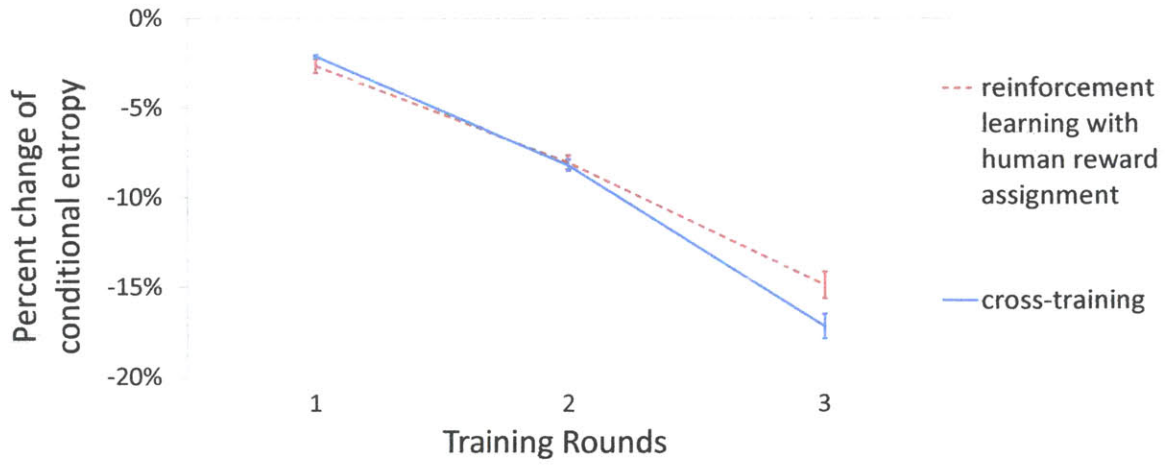

Figure 4-7: Human-Robot Mental Model Convergence. The graph shows the percent decrease of entropy rate over training rounds.

We noticed that the cross-training session lasted slightly longer than the reinforcement learning with human reward assignment session, as switching roles took more time on average than assigning a reward after each robot action. Since participants often interrupted training to interact with the experimenters, we were unable to reliably measure the training time for the two groups.

The above results support our first hypothesis: Cross-training improves quantitative measures of human-robot mental model convergence.

4.3.2 Qualitative Measures

After each training round, each participant was asked to rate his or her agreement with the following statement on a five-point Likert scale: "In this round, Abbie performed her role exactly according to my preference, drilling the screws at the right time and in the right sequence." Participants were also asked to respond to a survey upon completion of the experiment. Subjects who cross-trained and then executed the task with Abbie (Group A) selected a significantly higher mark on the Likert scale than those who trained with Abbie using the standard reinforcement learning method (Group B) for the following statements:

- "In this round, Abbie performed her role exactly according to my preference, drilling the screws at the right time and in the right sequence.":
  (For the final training round) Group A: 4.52 [SD=0.96]; Group B: 2.71 [SD=1.21]; $p < 0.01$

- "In the actual task execution, Abbie performed her role exactly according to my preference, drilling the screws at the right time and in the right sequence.":
  Group A: 4.74 [SD=0.45]; Group B: 3.12 [SD=1.45]; $p < 0.01$

- "I trusted Abbie to do the right thing at the right time.":
  Group A: 3.84 [SD=0.83]; Group B: 2.82 [SD=1.01]; $p < 0.01$

- "Abbie is trustworthy.":
  Group A: 4.05 [SD=0.71]; Group B: 3.00 [SD=0.93]; $p < 0.01$

- "Abbie does not understand how I am trying to execute the task.":
  Group A: 1.89 [SD=0.88]; Group B: 3.24 [SD=0.97]; $p < 0.01$
- "Abbie perceives accurately what my preferences are.":
  Group A: 4.16 [SD=0.76]; Group B: 2.76 [SD=1.03]; $p < 0.01$

The p-values above are computed for a two-tailed Mann-Whitney-Wilcoxon test. The results show that participants in Group A agreed more strongly that Abbie had learned their preferences, compared to those in Group B. Furthermore, cross-training had a positive impact on their trust in Abbie, in accordance with prior work [49]. This supports Hypothesis 2 of Section 4.2.1. The two groups did not differ significantly when subjects were asked whether they themselves were "responsible for most of the things that the team did well on this task," whether they were "comfortable working in close proximity with Abbie" or whether they and Abbie "were working toward mutually agreed upon goals."

4.3.3 Fluency Metrics on Task Execution

We elicited the fluency of the teamwork by measuring the concurrent motion of the human and robot and the human idle time during the task execution phase, as proposed in [23]. The measurements of the above metrics were evaluated by an independent analyst who did not know the purposes of the experiment, nor whether a participant had been a member of Group A or B. Additionally, we automatically computed the robot idle time and human-robot distance. Since these metrics are affected by the human's preferred way of performing the task, we used only the subset of participants who self-reported their preferred strategy as "while Abbie is drilling a screw, I will place the next one." This subset consisted of 20 participants and was the largest subset of participants who reported the same preference on task execution.

Concurrent Motion

We measured the time duration in which both human and robot were concurrently in motion during the task execution phase, and found that participants in Group A who

preferred to "finish the task as fast as possible, placing a screw while Abbie was drilling the previous one" had a 71% increase in the time of concurrent motion with the robot compared to participants in Group B who reported the same preference (A: 5.44 sec [SD = 1.13 sec]; B: 3.18 sec [SD = 2.15 sec]; $p = 0.02$). One possible explanation for this difference is that cross-training engendered more trust in the robot (supported by subjective results presented in Section 4.3.2), and therefore participants in Group A had more confidence to act while the robot was moving.

Human Idle Time

We measured the amount of time each human spent waiting for the robot. Participants in Group A spent 41% less time idling, on average, than those in Group B a statistically significant difference (A: 7.19 sec [SD = 1.71 sec]; B: 10.17 sec [SD = 3.32 sec]; $p = 0.04$). In some cases, the increase in idle time occurred because the participant was waiting to see what the robot would do next. In other cases, the robot had not correctly learned the human preference and did not act appropriately, confusing the human team-member or forcing them to wait.

Robot Idle Time

Our task-execution software automatically calculated the time that the robot remained idle while waiting for the human to perform an action, such as place a screw. The difference in idle time between Group A and Group B was statistically significant (A: 4.61 sec [SD = 1.97 sec]; B: 9.22 sec [SD = 5.07 sec]; $p = 0.04$).

Human-Robot Distance

Statistically significant differences between Group A and Group B were observed for the distance from the human hand to the robot base, averaged over the time the robot spent moving and normalized to the baseline distance from the participant (A: 23 mm [SD = 26 mm]; B: 80 mm [SD = 73 mm]; $p = 0.03$). This difference occurred because some participants of Group B "stood back" while the robot was moving. Previous work using physiological measures has shown that mental strain

among operators is strongly correlated with the distance of a human worker from an industrial manipulator moving at high-speed [4]. We therefore suggest that cross-training with the robot may have a positive impact on emotional aspects such as fear, surprise and tension, and leave further investigation to be conducted in future studies.

The above results confirm our third hypothesis: that human-robot interactive planning with cross-training improves team fluency metrics on task execution, compared to human-robot interactive planning using reinforcement learning with human reward assignment.

4.3.4 Transfer of Learning Experience from Virtual to Actual Environment

The significant differences in team fluency metrics and subjective measures between the two groups are indicative of a transfer of the learning experience from the virtual environment to the actual environment. In fact, we observed an intermediate correlation between the mental-model similarity metric, elicited after the training process, and the time of human-robot concurrent motion at task execution ($r = 0.37$). Additionally, we found an intermediate correlation between the entropy-rate after the final training round and the concurrent motion ($r = 0.59$), as well as human idle time ($r = -0.69$) during task execution. Finally, an intermediate correlation was observed between the entropy-rate and the participants Likert-scale response to the statement: "In the actual task execution, Abbie performed her role exactly according to my preference, drilling the screws at the right time and in the right sequence" ($r = 0.49$). While these results do not fully support our fourth hypothesis, they are indicative of a transfer of learning from virtual to actual environment, and warrant further investigation.

4.4 Post-hoc Experimental Analysis

In this section, we use the data collected from the human subject experiment to discuss the entropy-rate as a method to dynamically assess change in human preference or a human mistake. A change in human preference during task execution could mean, for instance, that a new user has arrived, and therefore the new human and robot should cross-train before performing the task. Dynamic detection of human mistakes could serve as an automated inspection mechanism to encourage the human to self-correct, while increased detection of inconsistencies in human actions could be a sign of fatigue. There is great potential for robots to use this information to improve team efficiency and safety: For instance, the robot could adapt its action selection and motion generation to avoid areas where there is greater uncertainty about human behavior.

We support this assertion with the entropy-rate plots of three participants who changed their screw placement sequences during training. Additionally, we conduct an analysis of the algorithmic performance of cross-training, and explain why it outperformed SARSA(λ) in the human subject experiment.

4.4.1 Dynamic Error Detection Using Entropy Rate

We discuss the entropy-rate as a method for dynamic error detection. First, we explain the reason for the entropy rate sensitivity in the human strategy, and then show the evolution of the entropy rate for two participants in Group A and one participant in Group B who changed their strategy during task execution.

After the training session, we asked all participants to annotate their preferred sequences of human and robot actions toward task completion. We calculated the entropy rate using Eq. 3.1 of Section 3.3, taking into account only the states that appeared in the annotated sequence. As human and robot follow a mutually agreed-upon sequence of actions during training, the uncertainty that the robot has about the human actions in these states decreases. On the other hand, if the human changes the sequence of their screw placement at some point, the transition probability dis-

tribution over the next states becomes flatter, and the entropy increases. We would like to note, however, that this increase appears only when the robot has correctly learned the users preference to some degree. For instance, if the user annotates as their preferred sequence to "have a screw drilled as soon as it is placed in the order A-B-C", the states in the annotated sequence used for the calculation of the entropy-rate are: "no screw placed," "screw A placed, "screw A drilled and screw B placed," etc. However, if the robot has not yet learned that it should drill after the user places a screw, most of these states are not reached, and therefore any change in the placement sequence will not affect the entropy calculation. We present three examples of participants who changed their strategy while working with the robot.

1. Subject 1, Group A: This user's stated preference was to "place the screws down in the order B-A-C, and Abbie drills them immediately after each one is placed." The user followed this preference during the first two rounds, but changed the sequence from B-A-C to A-C-B during the third round, causing an increase in the entropy rate. At task execution, the user then switched back to the predefined sequence B-A-C, and the entropy decreased again (Figure 4-8).

2. Subject 2, Group A: This participant followed her initial stated preference of placing the screws in the order C-B-A during training, but switched to the sequence A-B-C during task execution without realizing it. The robot had correctly learned her preference of C-B-A during training, and the result of the change of strategy at execution was a sharp entropy increase, as illustrated in Figure 4-9.

3. Subject 3, Group B: This user started with a stated preference of placing screws in the order C-B-A, with Abbie "drilling them as they are put in place." During the first round, the robot did not learn the users preference, and instead waited for the user to finish placing all screws. At the second round, the participant changed the sequence to A-B-C. However, this change affected states that were not included in entropy-rate calculation, as explained at the beginning of

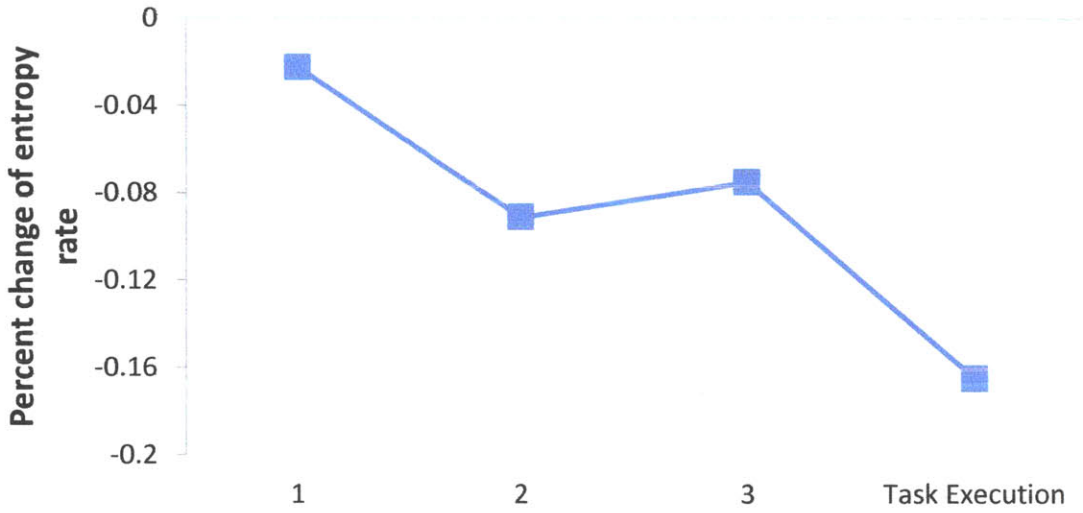

Figure 4-8: Entropy-rate of subject 1. The change in the participant's strategy is illustrated by an increase in the entropy-rate at the third round.

the session, and therefore entropy remained constant during the second round (Figure 4-10).

In conclusion, we observed an increase in the entropy rate when there were changes or inconsistencies in execution, and when these changes occurred after the human and robot had converged to a mutually agreed-upon sequence of actions toward task completion. Practical use of this metric as an informative measure at task execution would require confirmation that the robot had correctly learned the humans preference; this confirmation could be obtained by the human upon completion of training. After each task execution, the entropy-rate decrease can be compared to that of a consistent user via a distance metric, and a large deviation can signify a change in human behavior. We leave the testing of this hypothesis for future investigation.

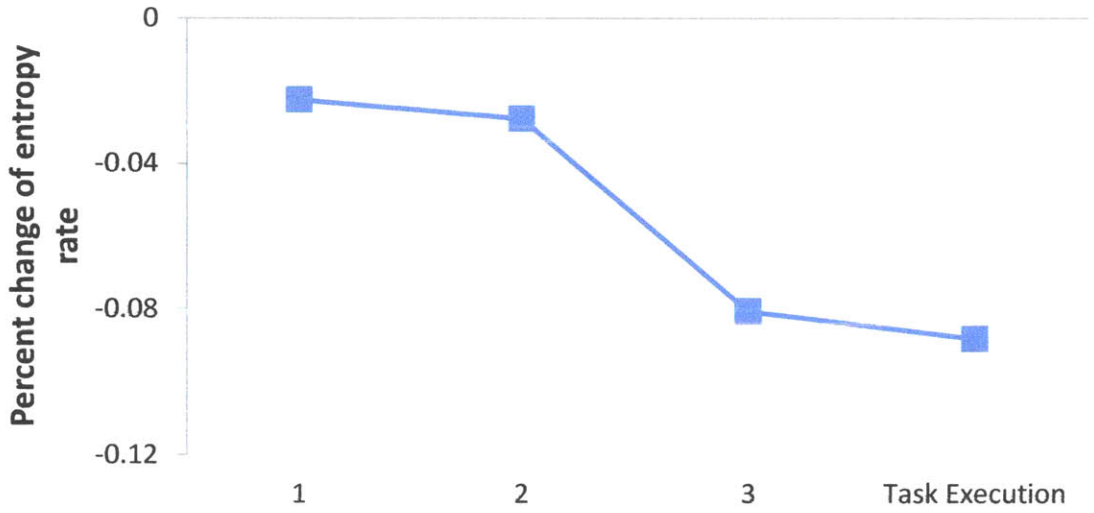

Figure 4-9: Entropy-rate of subject 2. The change in the participant's strategy is illustrated by an increase in the entropy-rate at task execution.

4.4.2 Algorithmic Performance

The results presented in Section 4.3 imply that the robot better learned human preferences through cross-training than training using reward assignment. Upon analysis of the experimental data, we identified three main reasons for this difference:

First, if the robot performs an action that does not match the humans preference, the human will then assign a negative reward, and the SARSA(λ) algorithm will update the value of the corresponding state-action pair that estimates the expected return. When the same state is visited again, the algorithm will most likely not result in the same action, as its value has been reduced. However, the robot will not have any information about which of the other available actions best matches the preference of the human. On the other hand, in the cross-training algorithm, when the human switches roles with the robot, he directly demonstrates his preferred robot action, and the rewards of the visited states are updated. Therefore, the values of the most relevant states are affected to the greatest extent after each iteration,

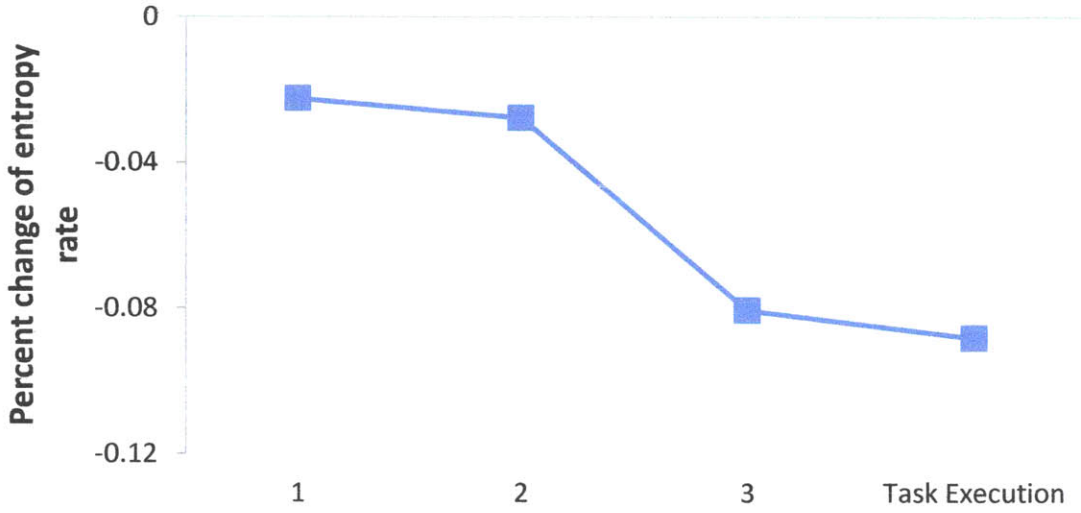

Figure 4-10: Entropy-rate of subject 3. The entropy-rate does not increase when a change in the sequence occurs in states irrelevant to the user preference.

speeding up the learning process for the robot. To verify the above, we calculated for each algorithm the ratio of the number of visited states during training that matched the human preference, as elicited after the training process, to the total size of the state-space. For participants in Group A, this ratio was 76% while for participants of Group B it was 67%, supporting our explanation.

Second, we observed that some participants in Group B had a tendency toward more neutral reward values, even when the actions performed by the robot were very different from their stated preferences. This slowed the performance of SARSA(λ) for these participants.

Third, even though we explicitly asked participants in Group B to evaluate the action that the robot performed after each state, some participants treated the reward as a future-directed signal. We will provide as an example one participant who preferred that the robot drill a screw as he was placing the next one, in a direction from left to right. During the first training round, the robot policy was initialized so that it was very different from human preference. Therefore, the human placed the

first screws and the robot waited instead of drilling, contrary to the stated preference. When the human finished placing all screws, the robot began drilling at the leftmost screw. The participant then assigned a positive reward to the robot, assuming that this would encourage the robot drilling behavior. This resulted in an increase in the estimated value of the state-action pair, "drill screw" at "screw A placed, screw B placed, screw C placed". However, the state "all screws have been placed" would never appear if the robot had followed the human preference of drilling a screw as soon as it was placed. By assigning a positive reward to the aforementioned state-action pair, the human increased the estimated value of a state that does not appear in his preferred sequence of states, and therefore misled the learning algorithm.

In conclusion, the proposed cross-training algorithm outperformed standard approaches with human reward assignment, as it enabled updating of the values of the most relevant parts of the state-space, and switching roles is more intuitive to a human participant compared to assigning rewards. However, reinforcement learning with human reward assignment has proven very effective when a human teacher guides an agent toward maximizing an objective performance metric [29]. We believe that the above observations are helpful in effectively designing the user interface and reward assignment method in such a case.

4.5 Conclusion

We designed and evaluated human-robot cross-training, a strategy widely used and validated for effective training in human teams. We first presented a computational formulation of the robot's teaming model and showed that it is quantitatively comparable to the human mental model. Based on this encoding, we formulated human-robot cross-training and evaluated it in a large-scale experiment of 36 subjects. We found that cross-training improved quantitative measures of human-robot mental model convergence ($p = 0.04$) and mental model similarity ($p < 0.01$), while post-hoc experimental analysis indicated that the proposed metric of mental model convergence could be used for dynamic human error detection. A post-experimental survey yielded

statistically significant differences between groups in perceived robot performance and trust in the robot ($p < 0.01$). Finally, we observed a significant improvement in team fluency metrics, including an increase of 71% in concurrent motion ($p = 0.02$) and a decrease of 41% in human idle time ($p = 0.04$), during the human-robot task execution phase in the cross-training group. These results provide the first evidence that human-robot teamwork is improved when a human and robot train together by switching roles in a manner similar to effective training practices for human teams.

In this experiment, we focused on a simple place-and-drill task as a proof of concept. We are currently extending the cross-training algorithm to a complex hand-finishing task, wherein the robot manipulator lifts and places a heavy load at an ergonomically friendly position for the human, whose role is to refinish the surfaces of the load. The best position and orientation of the load depend on the size of the human, his arm length, his age and other physical characteristics, and therefore should be different for each individual worker. Additionally, there is a wide variety of different potential preferences for the sequence of surface refinishing and the velocity of robot motion. As this task must be encoded in a very large state-space, we will need to use value-function approximation methods for the reward-update of the rotation phase, rather than the currently implemented tabular approach.

In the next chapter we extend the computational formulation of the robot's teaming model to a Partially Observable Markov Decision Process framework [25]. In the future, we plan to incorporate information-seeking behavior, and testing this framework using more complex tasks. Although cross-training is applicable to a wide range of manufacturing tasks with well-understood task procedures, there are also tasks that are difficult to model and simulate in a virtual environment, such as robot-assisted surgery. For these cases, other team training techniques, such as perturbation training as presented in Section 2.1, could be more suitable; however, we leave this assessment for future work.

Chapter 5

Efficient Model Learning for Human-Robot Collaborative Tasks

5.1 Introduction

New industrial robotic systems that operate in the same physical space as people highlight the emerging need for robots that can integrate seamlessly into human group dynamics, by adapting to the personalized style of the human teammates. This adaptation requires learning a statistical model of the human behavior and integrating this into the general decision making of the robot in a principled way. This work presents a framework on learning types of humans from observation of human teams, and associating each type with an action selection mechanism. The learning is done completely automatically, without any human intervention. Additionally, the robustness of the action selection mechanism of the robot is compared to previous model-learning algorithms for increasing deviations of the human actions from the demonstrated behavior.

This work is based on our observation that even where there is a large number of different human preferences, the actual high-level strategies followed by humans working in teams are generally limited in number. We use this insight to denote the preference of a human team member on his teammate as a partially observable variable in a Mixed Observability Markov Decision Process [44], and constrain its value

to a limited set of possible assignments. We chose the MOMDP formulation, as the number of observable variables in human-robot collaborative tasks in manufacturing settings is much larger than the partially observable ones. We define as human type the preference that the human has on a subset of the task-related actions taken by the robot, in a collaborative task. Denoting the human preference on the actions of his partner as a hidden variable naturally models collaboration with humans, since their intentions can never be directly observed in the training, and must be inferred through interaction and observation.

We present a framework for learning human user models from joint-action demonstrations, that enables the robot to compute a robust policy on a collaborative task with a human. We assume access to demonstrations of human teams working on a collaborative task. First, we describe the clustering of the demonstrated action sequences into different human types using an unsupervised learning algorithm. We then use the demonstrated sequences to learn a reward function that is representative for each type, by employing an inverse reinforcement learning algorithm. The learned model is then used as part of a MOMDP formulation, wherein the human type is a partially observable variable. With this framework, we can infer either offline or online the type of a new human user that was not included in the training set and we can compute a policy for the robot that will be aligned to the human preference and will be robust to deviations of the human actions.

5.2 Relevant Work

Learning a model of a human usually requires a human expert to explicitly teach the robot a skill or a specific task [5, 6, 2, 42, 15, 3]. In manufacturing settings, a large part of work is done manually by humans. Wherever we see manual work, people develop their own personalized style of doing the task, although some aspects of the task at hand are well-defined. In this work, we use demonstrations of human teams executing a task, and automatically learn the human types using unsupervised learning. The data of each cluster is then inputted to 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 [11]. There have been a number of inverse reinforcement learning methods developed, many of which use a weighted-features representation for the unknown reward function. We follow the approach of Abbeel and Ng [2], solving 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. Other approaches find a weight vector that explains the expert demonstrations by optimizing the margin between competing explanations. There have also been game-theoretic approaches [52, 56] that aim in modeling multi-agent behavior. We use human demonstrations to learn a number of different human types and a reward function for each type, and use these as part of a MOMDP formulation.

Related approaches in learning user models include natural language interaction with a robot wheelchair [17], where a user model is learned simultaneously with a dialog manager policy. The human interacts with the system by giving verbal commands, as well as a scalar reward after each robot action. The model is encoded in the transition functions, observation functions and rewards of a Partially Observable Markov Decision Process framework. The system initially assumes that the model parameters are initially uncertain, and improves the model through interaction. Rather than learning a new model for each human user, which can be tedious and time-consuming, we use demonstrations from human teams to infer some "dominant" human types and then associate each new user to the new type. For pursuit games, researchers employ an empirical approach in which an agent plans using Monte Carlo Tree Search using a set of known models of possible teammates, which are then used to generate action likelihoods to infer the teammates type from observed behavior [8]. Rather than having a fixed set of known models, we estimate the models automatically from training data. Additionally, the robot uses observations, rather than states in its histories, which allows it to account for unobserved state variables in its plans.

Recent work also infers human intentions in collaborative tasks for game AI applications. [41] focus on the case of inferring the intentions of a human player, allowing a

Non-Player Character (NPC) to assist the human. They propose the CAPIR framework, in which a task is decomposed into subtasks, each of which is computationally tractable and is modelled by a Markov Decision Process. Alternatively, [32] proposed the Partially Observable Monte-Carlo cooperative planning system, in which the human intention is inferred for the cops-and-robbers turn-based game. The algorithm uses a black-box simulator to generate human actions, and interfaces it with a Monte-Carlo planner [50]. In both works, the model of the human type is assumed to be known beforehand.

Partially Observable Markov Decision Process models have been used to infer human intention on driving tasks [12] as well. Since the hidden variable is the human intention on its own actions, rather than the robot actions, the user model is represented by the transition matrix of a POMDP, instead of the reward structure. The transition matrix is learned using action-rules, that are task-specific. In our framework, none of the learning steps requires task-specific rules. Alternative POMDP models of multi-agent collaboration have been used for interactive assistant applications[19]. The MOMDP formulation [44] has been shown to achieve significant computational efficiency, and it has been used in motion planning applications [7], with uncertainty on the human intention over its own actions. In the aforementioned work, the reward structure of the task is assumed to be known. We automatically learn the reward function that corresponds to each human type is learned automatically from unlabeled demonstrations.

In summary, the proposed framework makes the following contributions.

- It enables fast estimation of a human user model, which can be done either offline or online, by learning a priori a set of "dominant" models using unsupervised learning. This differs from previous approaches [17] that start with uncertain model parameters and learn them through interaction. Although such approaches do not have the limitation of a fixed set of available models, they require a very large amount of data for a good model, which can be an issue when using them for practical applications.

- It uses a MOMDP formulation to compute personalized policies for the robot that take into consideration the uncertainty over the human type. Similar MOMDP formulations have been used in prior work [44], [7], but the reward structure there was assumed to be known. We present a pipeline to automatically learn the reward function of the MOMDP using unsupervised learning and inverse reinforcement learning. Research on POMDP formulations for collaborative tasks in game AI applications [41, 32, 50] also assumed a known human model.

- It presents a MOMDP formulation with a human type as the partially observable variable, and the reward function as a function of the human type. This allows the computation of a policy that is in accordance with the preference of the human teammate over what the robot should do. Previous partially observable formalisms [44, 7, 12, 19, 41, 32] in assistive or collaborative tasks had as partially variable the human preference or intention over its own actions, rather than the robot actions.

- It addresses the problem of seamless integration of a robot into human group dynamics, by adapting to the personalized style of the human teammate. We use data from actual human subject experiments to show that that the learned MOMDP policies perform close to the ones from a handcoded model from a domain expert, and significantly more robustly compared to previous algorithms for human-robot collaborative tasks that reason in state-histories [43].

We describe the proposed framework in the next section.

5.3 Method

Our proposed framework has two main stages, as shown in Figure 5-1. In the first stage the training data is preprocessed, and in the second stage the robot infers the personalized style of a new human teammate and executes its role on the task based on the teammate's preference.

Assuming access to a set of demonstrated sequences of alternating actions from human teams working together on a collaborative task, we use an unsupervised learning algorithm to cluster the data into dominating human types. We associate each type with a set of sequences of alternating actions from human teams. When a robot is introduced to work with a new human worker, it needs to infer the human type and choose actions that are aligned to the preference of the human. Additionally, the robot should reason over the uncertainty on the type of the human. Therefore, the cluster indices are used as the values of a partially observable variable denoting the human type, in a Mixed-Observability Markov Decision Process. We learn a reward function for each human type, which represents the preference of the human of that type on a subset of task-related robot actions. We then compute an approximately optimal policy for the robot that maximizes the expected accumulated reward, reasoning over the uncertainty on the human type.

In the second stage, a new human subject is asked to execute the collaborative task with the robot. The human is first asked to demonstrate a few sequences of human and robot action. A belief over his type is then computed based on the likelihood of the human sequences belonging to each cluster. Alternatively, if the human actions are informative on his type, that is his preference on the robot actions, the human type can be estimated online. The robot then executes at each timestep the action based on the computed policy of the MOMDP, based on the current belief of the human type.

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

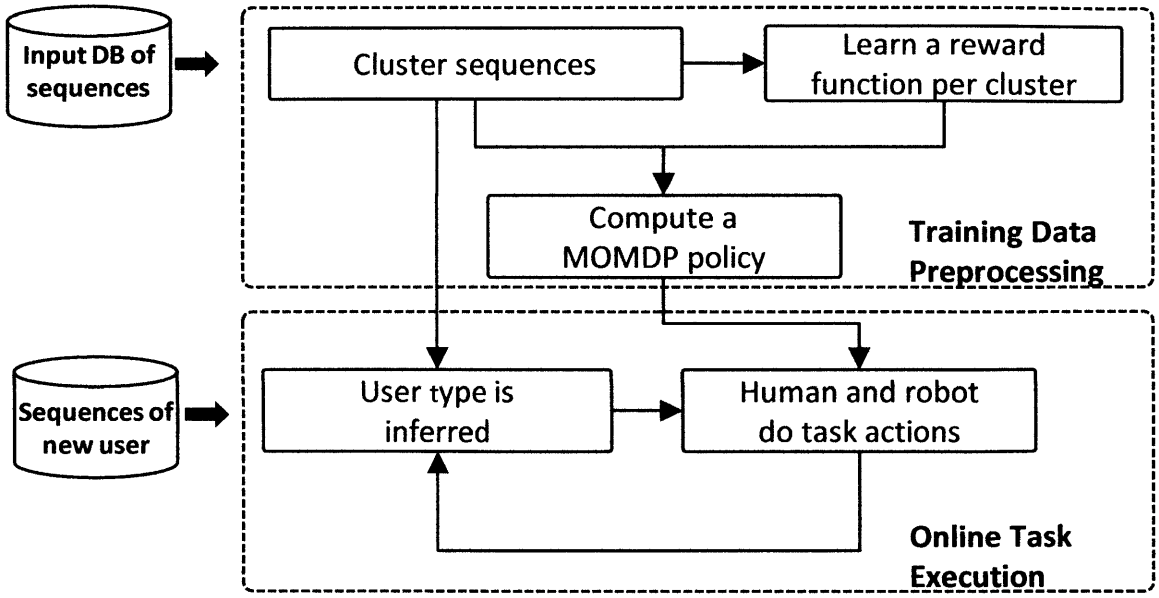

Figure 5-1: Framework flowchart

5.4 Clustering of Human Types[1]

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 actions that the human would like the robot to take. We can determine these actions, for example, by observing how two humans work together. The goal is to find the number of human types, k, in 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 using various methods. Murphy and Martin [38] 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 [33]. To select the best model, Bayesian information criterion (BIC) and integrated complete likelihood (ICL) were used. In another work, Jääskinen [24] clustered DNA

[1]This work was done by Ramya Ramakrishnan. It is reproduced in this thesis for completeness of the framework

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 of these two methods. Similar to [24], we learn transition matrices between human and robot actions using EM because this would provide information about how the human will act based on the robot and visa versa. However, we use a uniform prior distribution over the partitions, rather than the Dirichlet process prior [24], as it was sufficient for our task. We use BIC to find the ideal value of k, as done in [38], rather than the greedy approach in [24], due to the small number of possible values of k. Again, we base this on the observation that even in complex tasks, the actual high-level strategies followed by humans working in teams are usually few in number.

We begin by using a hard variant of EM, similar to [24], 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 where the human and robot take turns, the transition matrix encodes information about how the human will act based on the previous robot action and visa versa.

We 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 $D = x_1...x_n$ needs to 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 $z_i \in \{1, ..., k\}$.

The probability of one sequence x_i parametrized by $\boldsymbol{\theta}$ can be represented by:

$$\begin{aligned} P(x_i; \boldsymbol{\theta}) &= \sum_{z_i=1}^{k} P(z_i) P(x_i | z_i; \boldsymbol{\theta}) \\ &= \sum_{z_i=1}^{k} P(z_i) \left(\prod_{j=2}^{l} \theta_{z_i}(x_i^j | x_i^{j-1}) \right) \end{aligned} \tag{5.1}$$

Algorithm: Clustering-Transition-Matrices-using-EM (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 θ_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 5-2: Clustering Transition Matrices using EM

x_i^j denotes the j^{th} element of the i^{th} demonstrated sequence.

For all data points, the log-likelihood can be represented by:

$$\begin{aligned} l(D; \boldsymbol{\theta}) &= \sum_{i=1}^{n} log P(x_i; \boldsymbol{\theta}) \\ &= \sum_{z=1}^{k} \sum_{i=1}^{n} \delta(z|z_i) log \left(P(z_i) \prod_{j=2}^{l} \theta_{z_i}(x_i^j | x_i^{j-1}) \right) \end{aligned} \tag{5.2}$$

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

In the Clustering-Transition-Matrices-using-EM algorithm, we learn the optimal transition matrices $\hat{\theta}_1, ..., \hat{\theta}_k$ by iteratively performing the E-step and the M-step. We first randomly initialize k transition matrices and sequence assignments (lines 1-2). We then repeatedly execute the E-step and M-step until the assignments Z

Algorithm: Select-Best-Model (k_{min}, k_{max}, $numOfIterations$)

1. **for** $k = k_{min}$ **to** k_{max}
2. **for** $i = 0$ **to** $numOfIterations$
3. Call Clustering-Transition-Matrices-using-EM (k)
4. Calculate the log-likelihood for this model:
   $l(D; \hat{\boldsymbol{\theta}}) =$
   $$\sum_{z=1}^{k} \sum_{i=1}^{n} \delta(z|z_i) log \left(P(z_i) \prod_{j=2}^{l} \theta_{z_i}(x_i^j | x_i^{j-1}) \right)$$
5. Calculate BIC term for this value of k:
   $BIC = l(D; \hat{\boldsymbol{\theta}}) - \frac{K}{2} log(n)$
   where K is the number of parameters
   and n is the number of data points.
6. For the current value of k, choose the cluster partition with the highest BIC value.
7. Return the value of k with the maximum BIC value and the corresponding cluster partition.

Figure 5-3: Finding Ideal Number of Clusters using BIC

have converged to stable values (lines 3, 6). In the E-step, we complete the data by assigning each sequence to the cluster that has 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).

The EM algorithm used here, however, requires k as input, but since we use an unsupervised clustering approach, this value is unknown to us. We run the Select-Best-Model algorithm to find the ideal value of k using Bayesian information criterion (BIC). As input, we specify the range of possible values for k: $k_{min} - k_{max}$ and run EM for each value of k within this range. For our problem, we chose the range for k to be from 2 to 10, which is based on the observation that there tends to be only a few high-level human preferences for a particular task (line 1). In addition to testing multiple values of k, because the results can differ based on initialization and EM often finds locally optimal solutions, for each value of k, we run EM for multiple iterations, specified by the input $numOfIterations$. In our case, we use $numOfIterations = 20$, as this was sufficient to see consistent results (line 2). After each run of EM, we calculate the log-likelihood based on the resulting cluster partition, as specified in line 4. We then use BIC to introduce a penalty term for complex models, so as the value of k increases, the penalty increases as well. In this case, the number of parameters K is $k|A|(|A| - 1)$, since we have k transition matrices, each of which has $|A|(|A| - 1)$ free parameters (line 5). The cluster partition with the highest BIC value over all the iterations was chosen as the best model for that particular value of k (line 6). Comparing the BIC values for each value of k then determines the final cluster partition (line 7). By using EM and BIC in this way, we have found both the number of clusters and the cluster partition for this data.

We then input the learned clusters into a Mixed-Observability Markov Decision Process, which treats the human type as a partially observable variable that can take a finite set of values. Each human type value is associated with a corresponding cluster. In the next section, we describe the MOMDP formulation, the learning of a reward function for each human type value and the computation of an approximately

optimal policy for the robot.

5.5 Mixed Observability Markov Decision Process Learning and Planning

The clusters of the 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 and choose actions that are aligned to the preference of the human. Additionally, the robot should reason over the uncertainty on the type of the human. Therefore, the cluster indices are used as the values of a partially observable variable denoting the human type, in a Mixed-Observability Markov Decision Process. We learn a reward function for each human type, which represents the preference of the human of that type on a subset of task-related robot actions. We then compute an approximately optimal policy for the robot that maximizes the expected accumulated reward, reasoning over the uncertainty on the human type.

We describe the MOMDP formulation, the learning of the reward function and the computation of an approximately optimal policy as follows.

5.5.1 MOMDP Formulation

We treat the unknown human type as a hidden variable in a Mixed-Observability Markov Decision Process (MOMDP), and have the robot choose actions based on the estimated human type. The MOMDP framework uses proper factorization of the observable and unobservable state variables, which reduces 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:

- $\boldsymbol{X}$ is the set of observable variables in the MOMDP. In our framework, observable variable is the current task-step, among a finite set of task steps that signify the progress towards task completion.

- $\boldsymbol{Y}$ 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 that consist of the observable and non-observable variables. The state $s \in S$ consists of the task-step$x$, that 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 at the next time step being x' if robot takes action a_r from state s.

- $\mathcal{T}_y : S \times A_r \times X \longrightarrow \Pi(Y)$ is the probability of the hidden variable at the next time step being y' if robot takes action a_r from 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 the immediate reward for the robot taking an 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 from the game. An observation is the human and robot action taken.

- $O : S \times A_r \longrightarrow \Pi(\Omega)$ is the observation function, which gives for each state s, and robot action a_r, a probability distribution over possible observations. We write $O(s, a_r, o)$ the probability that we receive observation o given s and a_r.

5.5.2 Belief-State Estimation

Based on the above, the belief update is then [44]:

$$b_y(y') = \eta O(s', a_r, o) \sum_{y \in Y} \mathcal{T}_x(s, a_r, x') \mathcal{T}_y(s, a_r, s') b_y(y) \tag{5.3}$$

The denominator $Pr(o/a_r, b_y)$ can be treated as a normalizing factor, independent of s'. Note that in the proposed framework, the robot can estimate an unknown human type only if the human actions are informative on that type. That is, if the

human preference on the robot actions can be disambiguated by the human actions, then the robot will update its belief on the human type by Equation 5.3. If this is not the case, we ask the human of the unknown type to provide a sequence of demonstrations, and initialize the belief over the human types for that human by using Equation 5.1 of the Performance of Clustering of Human Types section.

5.5.3 Inverse Reinforcement Learning[2]

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 are aligned with the type of the human teammate, a reward function needs to 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 represents a significant barrier in 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 type.

For a fixed human type y, we can reduce the MOMDP into a Markov Decision Process (MDP). The Markov Decision Process in this context is a tuple (X, A_r, T_x, R, γ), where Y, 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 Markov Decision Process using the Inverse Reinforcement Learning (IRL) algorithm [2]. Note that we assume the human type to be constant in the demonstrated sequences. To compute the reward function for each cluster, we first assume there exists a feature vector φ for each state and a feature expectation for a given policy. A feature expectation over a policy represents the expected discounted accumulation of feature values based on the policy. Formally, we define the feature expectations of a policy π to be:

$$\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \varphi(s_t)|\pi] \tag{5.4}$$

[2]Joint work with Keren Gu

We 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:

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i=1}^{n_z} \sum_{t=0}^{\infty} \gamma^t \varphi(s_t^{(i)}) \tag{5.5}$$

The IRL Algorithm begins with a single random policy, and attempts to generate a policy that is a mixture of existing policies, and whose feature expectations are close to the ones of the policy followed by the expert. In our case, the "expert" demonstrations are the demonstrations followed by all humans of a particular type. The algorithm terminates when $||\mu_z - \mu(\tilde{\pi})||_2 \leq \epsilon$. The IRL algorithm is implemented as follows:

1. Randomly pick some policy $\pi^{(0)}$ and approximate via Monte Carlo $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.

2. Compute a new "guess" of the reward function by solving the following convex (quadratic) programming problem:

$$\min_{\lambda,\mu} ||\hat{\mu}_z - \mu||_2 \tag{5.6}$$

   subject to $\sum_{j=0}^{i-1} \lambda_j \mu^{(j)} = \mu$, $\lambda \geq 0$, and $\sum_{j=0}^{i-1} \lambda_j = 1$. We set $t^{(i)} = ||\hat{\mu}_z - \mu||_2$ and $w^{(i)} = \frac{\hat{\mu}_z - \mu}{||\hat{\mu}_z - \mu||_2}$.

3. If $t^{(i)} \leq \epsilon$, then terminate.

4. Use reinforcement learning to compute the optimal policy $\pi^{(i)}$ of the MDP using the reward function $R(s) = w^{(i)}\phi(s)$.

5. Approximate via Monte Carlo $\mu^{(i)} = \mu(\pi^{(i)})$.

6. Set $i = i + 1$, and go back to step 2.

The result of the IRL algorithm is a list of policies $\{\pi^{(i)} : i = 0 \ldots M\}$ with mixture λ_i and feature counts μ. M is the total number of iterations. Each policy i maximizes

the expected accumulated reward function, calculated as $R(s) = w^{(i)}\phi(s)$. The output of the algorithm is a reward function, computed by taking the average of the weight values $w^{(i)}$ over the second half of the iterations. We ignore the first half in the averaging, as the first policies (and associated weights) generated by the algorithm are of lesser quality.

For each human type, we run the inverse reinforcement learning using as input the demonstrated sequences of that type to calculate an associated reward function. Having 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. This is described in the next section.

5.5.4 Policy Computation

We can solve for a policy that will take into account the uncertainty of the robot over the human type y. A standard POMDP solution maximizes the expected reward for each belief. Using the concept of a value function to represent a policy, we let the value function $V(x, b_y)$ represent the expected reward of a MOMDP agent starting with belief (x, b_y). The optimal value function is unique and satisfies the Bellman equation:

$$\begin{aligned} V(x, b_y) &= \max_{a_r \in A_r} Q(x, b_y, a_r) \\ Q(x, b_y, a_r) &= R(x, b_y, a_r) + \gamma \sum_{o \in \Omega} O(o|x, b_y, a_r) V(x, b^o_{a_r}) \end{aligned} \tag{5.7}$$

The belief $b^o_{a_r}$ is the belief after a Bayesian update using Equation 5.3. Equation 5.7 may be solved iteratively. Each iteration, or *backup*, brings the value function closer to its optimal value. Once the value function has been computed, it is used to choose actions. After each observation, we update the belief using Equation 5.3. The next action is then chosen using $\text{argmax}_{a_r \in A_r} Q(x, b_y, a_r)$, with $Q(x, b_y, a_r)$ given in Equation 5.7.

An exact solution to Equation 5.7 using an iterative backup approach is exponentially expensive. Instead, we can select a set of belief points, and then track the value

and its derivative for those belief points. The quality of approximation is determined by the choice of belief points. To calculate the optimal policy, we used the SARSOP algorithm, due to its ability to scale-up to hundrends of thousands of states [30, 7]. The SARSOP algorithm samples a representative set of points from the belief space that are reachable from the initial belief, and uses it as an approximate representation of the space, allowing for an efficient computation of a satisfactory solution.

5.6 Evaluation

We show the applicability of the proposed framework on a place-and-drill collaborative task between a human and a robot.

We used data from a human subject experiment, where 18 human subjects provided demonstrations for a shared-location joint-action collaborative task. The human's role was to place screws in one of the three available positions. The robot's role was to drill each screw. The demonstrations were provided through a training phase, in which human and robot switched roles, giving the human the opportunity to show to the robot how he would like the task to be executed, by demonstrating robot drilling actions. To evaluate our framework, we used leave-one-out cross-validation, by removing one subject and using the demonstrated sequences from the remaining 17 subjects as the training set. We apply the clustering algorithm presented in the section Performance of Clustering of Human Types. In all cross-validation iterations, the human subjects were clustered into two types. A "safe" type, in which each screw was placed first before drilling, and an "efficient" type, in which each screw was drilled immediately after placement. The order of the screws did not affect the clustering. Note that the human demonstrations are on the robot actions (drilling), rather than on the actions of the original role of the human (placing). For each type, we then used the inverse reinforcement learning algorithm to learn a reward function associated with that type. The number of types and associated reward function was passed as input to the MOMDP formulation.

Therefore, the unobservable variable y in the MOMDP formulation could take

two values, "safe" or "efficient". The observable state variable for this task x is the workbench configuration. The actions a_r are the drilling actions, as well as the noop. The human placing actions are encoded implicitly in the transition and observation matrices.

Each subject that was left out from the training set in the cross-validation, named as "testing subject", provided three demonstrated sequences of human and robot actions, using the same training phase, and a probability distribution over its type was calculated using Equation 5.1 of the Performance of Clustering of Human Types section. Using this as initial belief on the human type, and the associated reward function from the inverse reinforcement learning algorithm, the SARSOP solver computed a policy for the robot. We then had the testing subject execute the place-and-drill task with the actual robot, in a phase that we call "task execution phase" (Figure 5-4). In that phase, human and robot did their predefined roles, that is the human was placing screws and the robot was drilling screws. We assume that the preference of the testing subject over the robot actions is the same when demonstrating the actions and also when executing the task. This assumption was verified by the subject responses in a post-experimental questionnaire.

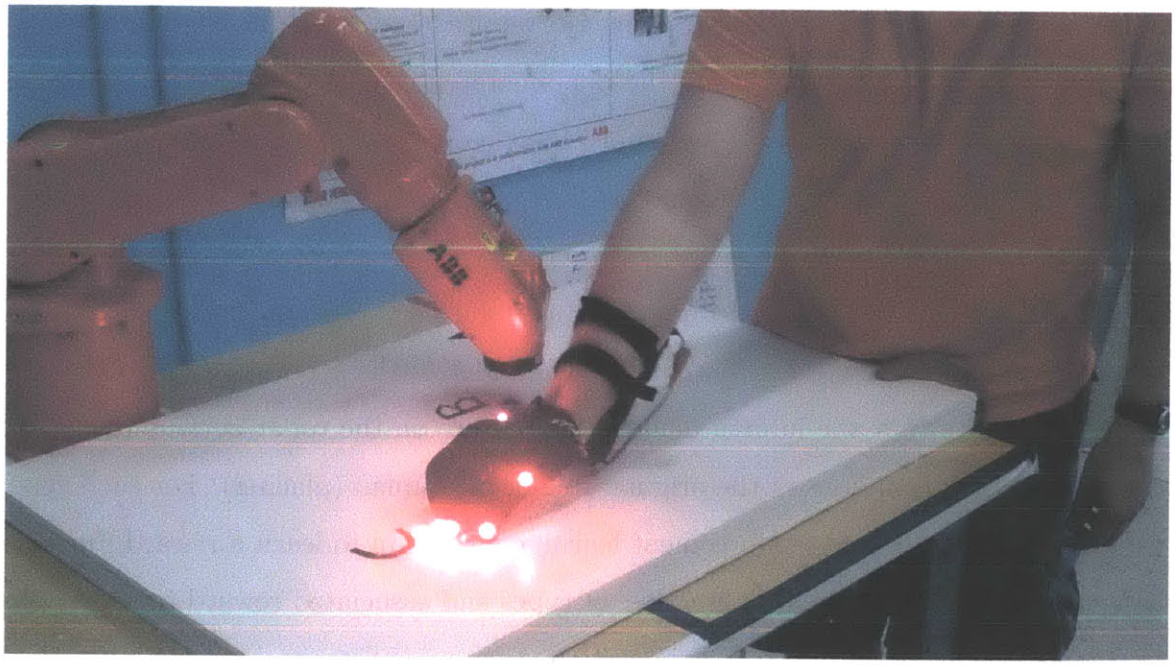

Figure 5-4: Task execution from a human-robot team on a place-and-drill task.

5.7 Performance of Clustering of Human Types[3]

Clustering of human types was performed on the place-and-drill data, which consisted of 54 demonstrated sequences, 3 for each of the 18 participants. We consider only the phase in the experiment where the human performs the robot actions and the robot performs the human actions because this reveals the human's preference over the robot actions. To evaluate whether the robot is acting as a human would, we compare the order of the place actions of the robot performing the human actions with the order of the human doing them, and we see that the actions match 87.65% of the time. This indicates that although the robot does not perform exactly as the human does, the actions are similar enough that we can consider these sequences in our clustering approach. We do not use the phase of the experiment where the human and robot are performing their respective roles because this does not capture human preferences over the robot actions.

To validate the clustering on this data, we used leave-one-out cross-validation, testing on each participant. Because each participant had 3 demonstrated sequences, we took a weighted average of the resulting assignments of the 3 sequences to determine the final type assigned to the participant. The weights were obtained from the likelihoods calculated in the EM algorithm. The type predicted by the clustering algorithm was then compared against manual labels handcoded by a human expert. We obtained an average accuracy of 96.5% for this data. This algorithm performs well on this data set because the data is simple and there is a clear partition in the sequences that separate the "safe" type from the "efficient" type. We believe, however, that this algorithm can be used for more complex data sets.

5.7.1 Robustness of Computed Policy

We compare the computed policy with a state-of-the-art iterative algorithm for human-robot collaborative tasks, called human-robot cross-training [43], in which the robot

[3]This work was done by Ramya Ramakrishnan. It is reproduced in this thesis for completeness of the evaluation section

learns a model of the human by switching roles. We use the demonstrated sequences of the testing subject as input to the cross-training algorithm, which learns a user model by updating the transition and reward function of a Markov Decision Process. The algorithm then computes a policy based on the learned model, which was shown to match the human preference in task execution, when human and robot resume their predefined roles [43].

In the actual human subject data, the human placing actions in task execution were in most cases identical to the ones provided in the demonstrations, therefore we simulate the task execution for increasing degrees of deviations from the demonstrated actions of the human. For instance, if in the demonstrated sequences the human placing actions were first "Place screw A", second "Place screw B", and finally "Place screw C", during the actual task execution we gradually increase the probability of the human choosing a different placing action at each task-step, leading the execution to previously unexplored parts of the state-space. We do this by having a simulated human do a random placing action with a probability ϵ, or the actual action taken by the testing human subject with probability $1-\epsilon$. The x-axis of Figure 5-5 denotes the value of ϵ. For increasing levels of deviations, we compute the accumulated reward for the policy of the proposed framework, and the policy computed by the human-robot cross-training algorithm. We do this for each iteration of the cross-validation, and plot the mean accumulated reward (Figure 5-5). We see that the policy of the human-robot cross-training algorithm performs similarly to the one of the proposed framework, if the user does not deviate from his demonstrated placing actions. However, as the deviations increase, the policy from the cross-training algorithm performs worse. On the other hand the MOMDP agent reasons over the partially observable human type, using a reward function that is learned from all demonstrated sequences that belong to the cluster associated with that type, and therefore its performance is not affected by these deviations.

5.7.2 Quality of Learned Model

To evaluate the quality of the clusters and corresponding reward functions that are generatede automatically from our framework, we had a domain expert partition manually the data and empirically handcraft a reward function. Figure 5-5 shows that the policy computed by the MOMDP using the automatically generated user model is comparable to the one that uses the handcoded model by the human expert. The plotted lines denote the accumulated reward, averaged over all iterations of cross-validation.

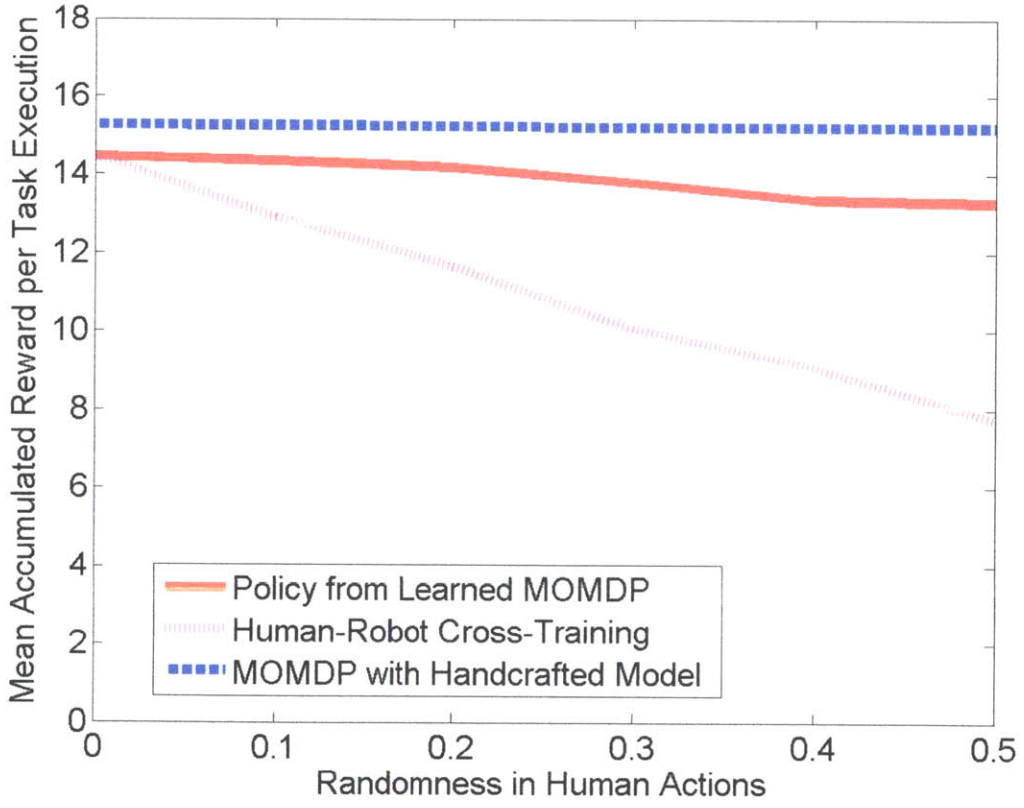

Figure 5-5: Accumulated reward averaged over 18 iterations of cross-validation, one for each human subject. The plotted lines illustrate the performance of a policy of a MOMDP model handcoded by a domain expert, the learned policy of the automatically generated MOMDP model using the proposed framework, and the learned policy from the Human-Robot Cross-Training algorithm. The x-axis represents the probability of the human taking a random action, instead of replaying the action he actually took in the task-execution phase with the robot. For each subject, we ran 100 simulated iterations of task execution.

5.8 Conclusion

We presented a framework that automatically learns the "dominant" types of human subjects, when working in teams on a collaborative task. Assuming access to a set of demonstrated sequences of alternating actions from human teams working together on a collaborative task, we find the number of human types by clustering these sequences. We then learn a user model for each type, represented by a reward function of a Mixed Observability Markov Decision Process. An approximately optimal policy that maximizes the expected accumulated reward is computed, taking into consideration the uncertainty on the human types. When a new human subject is introduced to execute the collaborative task with the robot, his type is inferred either offline from few demonstrations or online during task execution. Evaluation on a place-and-drill task shows that the robot performance is robust to increasing deviations of the human behavior from the demonstrated actions, compared to previous algorithms that reason in state-space. Futhermore, the performance is comparable to the policy of a MOMDP agent computed using a handcoded model by a domain expert. These results show that models of the human types in collaborative tasks can be efficiently learned and integrated into the general decision making, enabling robots to develop robust policies that are aligned with the personalized style of their human partners.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We designed and evaluated mechanisms for programming robots and training people in collaborative tasks.

First, we presented human-robot cross-training, a strategy widely used and validated for effective human team training. Cross-training is an interactive planning method in which a human and a robot iteratively switch roles to learn a shared plan for a collaborative task. We first presented a computational formulation of the robot's teaming model and show that it is quantitatively comparable to the human mental model. Based on this encoding, we formulated human-robot cross-training and evaluated it in a large-scale experiment of 36 subjects. We show that cross-training improves quantitative measures of human-robot mental model convergence ($p = 0.04$) and mental model similarity ($p < 0.01$). Post-hoc experimental analysis shows that the proposed metric of mental model convergence could be used for dynamic human error detection. Additionally, a post-experimental survey shows statistically significant differences in perceived robot performance and trust in the robot ($p < 0.01$). Finally, we observed a significant improvement in team fluency metrics, including an increase of 71% in concurrent motion ($p = 0.02$) and a decrease of 41% in human idle time ($p = 0.04$), during the human-robot task execution phase. These results provide the first evidence that human-robot teamwork is improved when a human

and robot train together by switching roles, in a manner similar to effective human team training practices.

Additionally, we presented a framework that robustly infers a human type from a fixed set of known human models from joint-action demonstrations. The robot then computes a policy that takes into account the uncertainty over the estimate of the human preference. We showed that the computed policy is robust to changes in the human behavior, and outperforms the policy learned from previous work.

6.2 Future Work

In the experiments of this thesis we focused on a simple place-and-drill task, as a proof of concept. We are currently extending the cross-training algorithm to a complex hand-finishing task, where the robot manipulator lifts and places a heavy load at an ergonomically friendly position for the human, whose role is to refinish the surfaces of the load. The best position and orientation of the load depends on the size of the human, his arm length, his age, and other physical characteristics, and therefore should be different for each human worker. Additionally, there is a wide variety of different preferences on the sequence of the surface refinishing, and velocity of robot motion. As this task needs to be encoded in a very large state-space, we will need to use value-function approximation methods for the reward-update of the rotation phase, rather than the currently implemented tabular approach.

Additionally, although cross-training is applicable to a wide range of manufacturing tasks, which have well-understood task procedures, there are tasks that are hard to model and simulate in a virtual environment, such as robot-assisted surgery. For these cases, other team training techniques, such as perturbation training, could be more suitable, and we leave this for future work.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[2] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*. ACM Press, 2004.

[3] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz. Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In *HRI*, pages 391–398, 2012.

[4] T. Arai, R. Kato, and M. Fujita. Assessment of operator stress induced by robot collaboration in assembly. *CIRP Annals - Manufacturing Technology*, 59(1):5 – 8, 2010.

[5] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[6] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, pages 12–20, 1997.

[7] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-aware motion planning. In *Algorithmic Foundations of Robotics X*, pages 475–491. Springer, 2013.

[8] Samuel Barrett, Peter Stone, and Sarit Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2011.

[9] Cannon-Bowers J.A. Blickensderfer E. and Salas E. Cross-training and team performance. *Making decisions under stress: Implications for individual and team training, American Psychological Association*, pages 299–311, 1998.

[10] Bruce Blumberg, Marc Downie, Yuri Ivanov, Matt Berlin, Michael Patrick Johnson, and Bill Tomlinson. Integrated learning for interactive synthetic characters. *ACM Trans. Graph.*, 21(3):417–426, July 2002.

[11] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[12] Frank Broz, Illah Nourbakhsh, and Reid Simmons. Designing pomdp models of socially situated tasks. In *RO-MAN, 2011 IEEE*, pages 39–46. IEEE, 2011.

[13] Blickensderfer E. Bowers C. Cannon-Bowers J.A., Salas E. The impact of cross-training and workload on team functioning: a replication and extension of initial findings. *Human Factors*, pages 92–101, 1998.

[14] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 233:1–233:8, New York, NY, USA, 2007. ACM.

[15] Sonia Chernova and Manuela Veloso. Teaching multi-robot coordination using demonstration of communication and state sharing. In *Proc. AAMAS*, Richland, SC, 2008.

[16] L. Tom Davis, Catherine D. Gaddy, John R. Turney, and Jennifer L. Koontz. Team skills training. *Performance + Instruction*, 25(8):12–17, 1986.

[17] Finale Doshi and Nicholas Roy. Efficient model learning for dialog management. In *Proc. HRI*, Washington, DC, March 2007.

[18] L. Ekroot and T.M. Cover. The entropy of markov trajectories. *Information Theory, IEEE Transactions on*, 39(4):1418 –1421, jul 1993.

[19] Alan Fern and Prasad Tadepalli. A computational decision theory for interactive assistants. In *Interactive Decision Theory and Game Theory*, 2010.

[20] Jamie C. Gorman, Nancy J. Cooke, and Polemnia G. Amazeen. Training adaptive teams. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 52(2):295–307, 2010.

[21] Jamie C Gorman, Nancy J Cooke, Harry K Pedersen, Jennifer Winner, Dee Andrews, and Polemnia G Amazeen. Changes in team composition after a break: Building adapative command-and-control teams. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 50, pages 487–491. SAGE Publications, 2006.

[22] G.R.J. Hockey, J. Sauer, and D.G. Wastell. Adaptability of training in simulated process control: Knowledge versus rule-based guidance under task changes and environmental stress. *Human Factors*, 49(1):158–74, 2007.

[23] Guy Hoffman and Cynthia Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *Proc. HRI*, pages 1–8, New York, NY, USA, 2007. ACM.

[24] Väinö Jääskinen, Ville Parkkinen, Lu Cheng, and Jukka Corander. Bayesian clustering of dna sequences using markov chains and a stochastic partition model. *Statistical applications in genetics and molecular biology*, pages 1–17, 2013.

[25] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134, May 1998.

[26] Frédéric Kaplan, Pierre-Yves Oudeyer, Enikö Kubinyi, and Adám Miklósi. Robotic clicker training. *Robotics and Autonomous Systems*, 38(3-4):197–206, 2002.

[27] W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proc. K-CAP*, September 2009.

[28] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proc. AAMAS*, May 2010.

[29] W. Bradley Knox and Peter Stone. Reinforcement learning from simultaneous human and mdp reward. In *Proc. AAMAS*, June 2012.

[30] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, pages 65–72, 2008.

[31] Janice Langan-Fox, Sharon Code, and Kim Langfield-Smith. Team mental models: Techniques, methods, and analytic approaches. *Human Factors*, 42(2):242–271, 2000.

[32] Owen Macindoe, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Pomcop: Belief space planning for sidekicks in cooperative games. In *AIIDE*, 2012.

[33] John I Marden. *Analyzing and modeling rank data.* CRC Press, 1995.

[34] M.A. Marks, M.J. Sabella, C.S. Burke, and S.J. Zaccaro. The impact of cross-training on team effectiveness. *Journal of Applied Psychology*, pages 3–13, 2002.

[35] Michelle A. Marks, Stephen J. Zaccaro, and John E. Mathieu. Performance implications of leader briefings and team-interaction training for team adaptation to novel environments. *J Appl Psychol*, 85:971–986, 2000.

[36] John E. Mathieu, Tonia S. Heffner, Gerald F. Goodwin, Janis A. Cannon-Bowers, and Eduardo Salas. Scaling the quality of teammates' mental models: equifinality and normative comparisons. *Journal of Organizational Behavior*, 26(1):37–56, 2005.

[37] John E. Mathieu, Tonia S. Heffner, Gerald F. Goodwin, Eduardo Salas, and Janis A. Cannon-Bowers. The influence of shared mental models on team process and performance. *Journal of Applied Psychology*, 85(2):273–283, 2000.

[38] Thomas Brendan Murphy and Donal Martin. Mixtures of distance-based models for ranking data. *Computational statistics & data analysis*, 41(3):645–655, 2003.

[39] Nicolás Navarro, Cornelius Weber, and Stefan Wermter. Real-world reinforcement learning for autonomous humanoid robot charging in a home environment. In *Proc. TAROS*, pages 231–240, Berlin, Heidelberg, 2011. Springer-Verlag.

[40] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[41] Truong-Huy Dinh Nguyen, David Hsu, Wee Sun Lee, Tze-Yun Leong, Leslie Pack Kaelbling, Tomas Lozano-Perez, and Andrew Haydn Grant. Capir: Collaborative action planning with intention recognition. In *AIIDE*, 2011.

[42] Monica N. Nicolescu and Maja J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proc. AAMAS*, pages 241–248, 2003.

[43] Stefanos Nikolaidis and Julie Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, HRI '13, pages 33–40, Piscataway, NJ, USA, 2013. IEEE Press.

[44] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[45] Phasespace motion capture `http://www.phasespace.com`, 2012.

[46] Deepak Ramachandran and Rakesh Gupta. Smoothed sarsa: reinforcement learning for robot delivery tasks. In *Proc. ICRA*, pages 3327–3334, Piscataway, NJ, USA, 2009. IEEE Press.

[47] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.

[48] Richard A. Schmidt and Robert A. Bjork. New conceptualizations of practice: Common principles in three paradigms suggest new concepts for training. *Psychological Science*, 3(4):207–217, 1992.

[49] Julie Shah, James Wiken, Brian Williams, and Cynthia Breazeal. Improved human-robot team performance using chaski, a human-inspired plan execution system. In *Proc. HRI*, pages 29–36, New York, NY, USA, 2011. ACM.

[50] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.

[51] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.

[52] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2007.

[53] Ana C. Tenorio-Gonzalez, Eduardo F. Morales, and Luis Villaseñor Pineda. Dynamic reward shaping: training a robot by voice. In *Proc. IBERAMIA*, pages 483–492, Berlin, Heidelberg, 2010. Springer-Verlag.

[54] Andrea L. Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In *Proc. AAAI*, pages 1000–1005, 2006.

[55] Andrea Lockerd Thomaz, Guy Hoffman, and Cynthia Breazeal. C.: Real-time interactive reinforcement learning for robots. In *Proc. of AAAI Workshop on Human Comprehensible Machine Learning*, 2005.

[56] Kevin Waugh, Brian D. Ziebart, and J. Andrew (Drew) Bagnell. Computational rationalization: The inverse equilibrium problem. In *Proc. ICML*, June 2011.