Transfer Learning of Human Preferences for Proactive Robot Assistance in Assembly Tasks

Heramb Nemlekar
University of Southern California
Los Angeles, California, USA
nemlekar@usc.edu

Neel Dhanaraj
University of Southern California
Los Angeles, California, USA
dhanaraj@usc.edu

Angelos Guan
University of Southern California
Los Angeles, California, USA
angelosg@usc.edu

Satyandra K. Gupta
University of Southern California
Los Angeles, California, USA
guptask@usc.edu

Stefanos Nikolaidis
University of Southern California
Los Angeles, California, USA
nikolaid@usc.edu

ABSTRACT

We focus on enabling robots to proactively assist humans in assembly tasks by adapting to their preferred sequence of actions. Much work on robot adaptation requires human demonstrations of the task. However, human demonstrations of real-world assemblies can be tedious and time-consuming. Thus, we propose learning human preferences from demonstrations in a shorter, canonical task to predict user actions in the actual assembly task. The proposed system uses the preference model learned from the canonical task as a prior and updates the model through interaction when predictions are inaccurate. We evaluate the proposed system in simulated assembly tasks and in a real-world human-robot assembly study and we show that both transferring the preference model from the canonical task, as well as updating the model online, contribute to improved accuracy in human action prediction. This enables the robot to proactively assist users, significantly reduce their idle time, and improve their experience working with the robot, compared to a reactive robot.

CCS CONCEPTS

• **Computer systems organization** → **Robotic autonomy**.

KEYWORDS

human preference, transfer learning, proactive robot assistance

ACM Reference Format:
Heramb Nemlekar, Neel Dhanaraj, Angelos Guan, Satyandra K. Gupta, and Stefanos Nikolaidis. 2023. Transfer Learning of Human Preferences for Proactive Robot Assistance in Assembly Tasks. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction (HRI '23), March 13–16, 2023, Stockholm, Sweden.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3568162.3576965

1 INTRODUCTION

To effectively assist human workers in actual assembly tasks, robots need to predict the sequence in which users will perform their actions [18, 19, 22]. For example, if a robot expects that the user will assemble a specific part at the next step, the robot can proactively fetch that part from the storage and deliver it to the user to reduce the time for which the user remains idle [16].

Since each user can have a different way of performing a given assembly [40, 41], assistive robots must learn the individual preferences of their users to predict their actions accurately [14, 27]. Typically, user preferences for task execution are learned in the form of a policy [2, 35] or a reward function [32, 40, 44] given the demonstrations of users in the task. However, demonstrating the preferred sequence of actions can be tedious and time-consuming for users in real-world assembly tasks.

Moreover, we consider the setting where the user executes the actual assembly only once, as in customized assemblies (e.g., satellite assembly). Our goal is to improve the team performance and fluency in the single task execution, instead of repeated executions, by accurately predicting user actions. Therefore, we focus on the

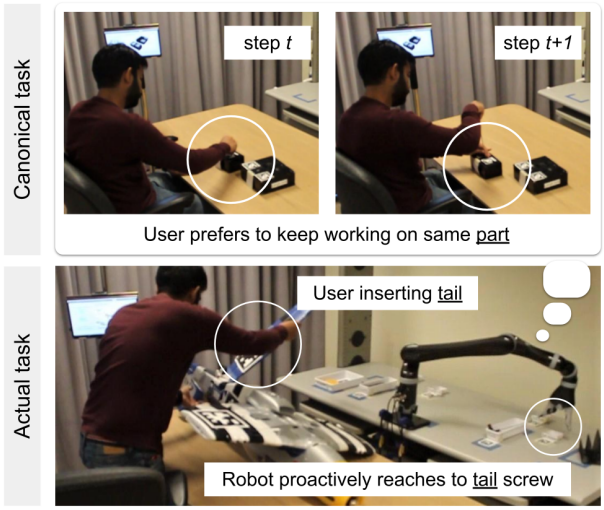

Figure 1: We use task-agnostic preferences in the canonical task, such as consecutively perform all actions that use the same part, as a prior for predicting user actions in the actual task. A robot uses the predictions to proactively assist users, while also updating the prior through interaction.

HRI '23, March 13–16, 2023, Stockholm, Sweden

ACM ISBN 978-1-4503-9964-7/23/03.
https://doi.org/10.1145/3568162.3576965

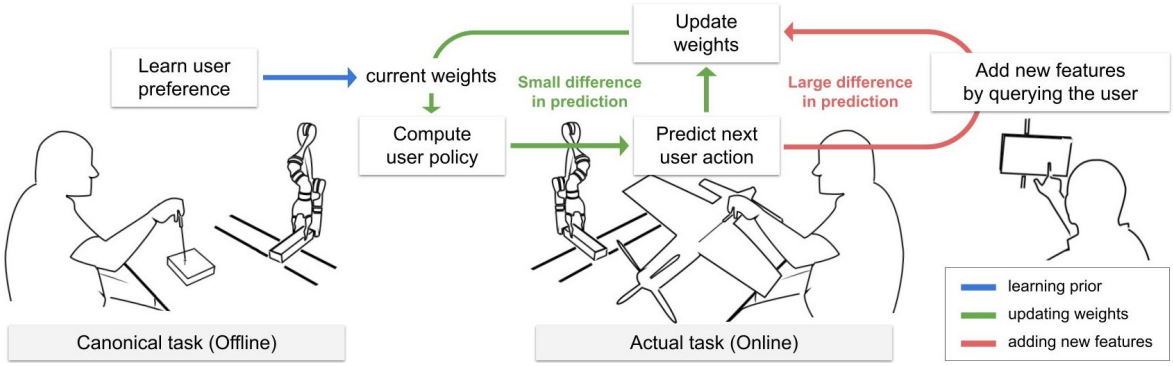

Figure 2: Proposed system for transfer learning of human preferences: In the offline phase, the robot learns the preference of a given user from their demonstrations in a canonical task. The preference is encoded as weights of task-agnostic features that constitute the user's reward function and is used as a prior estimate of their preference in the actual task (blue). In the online phase, the robot predicts the user's actions in the actual task based on its current estimate of their weights and updates them when the prediction is inaccurate (green). If the user's sequence is significantly different than the prediction, the robot asks users for new features to add to the reward function used to predict their actions (red).

problem of efficiently learning user preferences without obtaining their demonstrations in the actual task beforehand.

One approach is to learn dominant preference models by clustering demonstrations of previous users in the given task [29, 31] and then associate the preferences of new users with the learned models. While this removes the need for demonstrations from the new user, it still requires previous demonstrations on the task. Alternatively, users can observe robot actions and provide online corrections during task execution [5, 27]; however, an inaccurate initial preference model may result in a large number of interactions.

How can we obtain an accurate initial preference model without explicit demonstrations in the actual task? In a previous preliminary study [28], participants frequently explained their action selection in an assembly task using task-agnostic features. For example, some participants preferred leaving the actions requiring high physical effort towards the end of the task to avoid fatigue early on. Others preferred actions that allowed them to keep using the same part, rather than having to switch parts, regardless of the physical effort required for these actions. Therefore, a human preference model can be expressed with respect to task-agnostic features and learned from demonstrations in a much shorter, expressive task – called a *canonical* task – that allows users to demonstrate their preferences. In our example, the canonical task should include actions with varying physical efforts; it should also allow users to choose between keeping the same part and switching parts, so that we can learn a preference model with respect to these features.

However, transferring this model to the actual task is insufficient for making very accurate predictions. Human preferences often change between tasks because of human physical and mental states, such as fatigue and cognitive load [10]. Furthermore, users may consider new features that were not present in the canonical task, such as the space occupied by each part.

Our key insight is that *we can initialize the robot's preference model with demonstrations in a canonical task and then update the model through interaction in the actual task* (Fig. 2). Specifically, we model the user as maximizing a reward function represented by a weighted combination of task-agnostic features. Our system learns the feature weights from user demonstrations in the canonical task and uses them to predict user actions in the actual task. When the model predictions are inaccurate, we refine the preference model by (1) updating the weights to match the actual human actions or by (2) adding new features to the reward function after querying the user.

In summary, our main contribution is a novel system for predicting user actions in actual assembly tasks that combines transferring an initial preference model from a canonical task and updating the model online through interactions. We evaluate the proposed system in a user study, where participants demonstrate their preference in a canonical task and then perform a real-world model airplane assembly with a proactive robot. We show that our system results in accurate predictions, a significant reduction in human idle time, and an improvement in user experience, compared to a reactive robot. Our ablation studies show that both transferring the preference model from the canonical task and updating the model online are critical for accurate predictions. Finally, in a follow-up study, we show the benefit of adding new features when their preference cannot be accurately predicted with the existing set of features.

2 RELATED WORK

Learning user preferences offline. While user preferences can be modeled as constraints in the task scheduling problem [12], defining such constraints can be challenging for end users. Hence, preferences are often learned implicitly from demonstrations of the user's preferred behaviour [34, 43]. To efficiently infer user preferences, demonstrations of several users can be clustered to learn dominant preference models via inverse reinforcement learning (IRL) [1, 40], such that the preference of a new user can be quickly inferred by matching their actions to a dominant model [29, 31].

Since user demonstrations can be expensive to obtain in advance, an alternative is to obtain the user's preferred choice from a set of uniformly sampled [39] or actively generated trajectories [5]. While

trajectory comparisons help to fine-tune the estimated user preference, demonstrations offer a high-level initialization of the human's overall objective. Recent work has shown that initializing the belief over user preferences with just a solitary user demonstration can reduce the number of queries required to converge to the user's preference [5]. However, this would require the demonstrations to be obtained in the actual task prior to task execution.

To reduce the burden of obtaining demonstrations, previous work has explored transferring human strategies in simulated search-and-rescue tasks from simple to more complex environments [15]. Similarly, our preliminary work [28] has shown that in a user study without a robotic assistant, user preferences learned in a canonical task are useful for predicting human actions in an actual real-world assembly. The work assumes that user preferences do not change from the canonical to the actual task and that both the tasks share identical feature spaces. However, we show that the robot cannot solely rely on a transferred estimate of user preference, e.g., because of a change in preference or features that are not present in the canonical task. Hence, in this work, we propose a system that uses the preference model learned from the canonical task as a *prior* and updates it online through interaction.

Adapting to user preferences online. In the absence of pre-collected user demonstrations, robots can adapt to the changing user preferences by obtaining the user's feedback, e.g., the correct action to take in the current state, while executing the actual task. The dataset of user-approved state-action pairs can be used to learn a shaping function using regression trees that is added to the robot's existing quality function [27]. Alternatively, the dataset can be used to learn a feedback policy based on the number of right and wrong labels for each state-action pair and integrated with the robot's existing task policy [13]. Similarly, human interventions can also be used to directly update the parameters of the quality function via behaviour cloning [26].

To adapt the preference model while executing the first instance of the task, user feedback can also be recorded as physical corrections to the trajectories demonstrated by the robot [3, 21, 23]. The robot is initialized with a reward function that is a weighted combination of task features, where the weights are updated based on the difference in feature counts of demonstrated (by robot) and corrected trajectories (by human). User corrections can also be associated with a reward and used to update the robot's Q-function online [38]. Similarly, user feedback for right or wrong actions can be used as a reward signal for Q-learning [20, 24, 25].

Overall, when learning from human feedback, if the robot's initial estimate of user preference is inaccurate, the user will need to provide several corrections. Thus, the proposed system initializes the preference model with a prior learned from a canonical task to efficiently adapt to the user during the actual task execution.

3 SYSTEM FOR TRANSFER LEARNING OF HUMAN PREFERENCES

We model a task as a Markov Decision Process (MDP) defined by the tuple $M = (S, A^{\mathrm{H}}, T, R)$, where S is a finite set of states in the assembly, A^{H} is a finite set of discrete assembly actions that the user must perform to complete the assembly, $T(s_{t+1}|s_t, a_t^{\mathrm{H}})$ is the probability of transitioning from state s_t to s_{t+1} by taking action a_t^{H}, and $R(s_{t+1})$ is the reward received by the user in s_{t+1}. We assume that S, A^{H}, T are known, while R captures the (unknown) user preference. We also assume that the user maximizes their long-term expected reward on the task. Thus, given R, we can perform value iteration [4] to compute the user's policy $\pi(s_t)$ that maps s_t to a human action a_t^{H}. The robot can use $\pi(s_t)$ to predict the next human action and proactively assist the user for that action, e.g., by fetching the required part (section 5).

Algorithm 1 Online Update of Transferred Preference

Require: Transferred weights w_C, task-agnostic features ϕ, ϕ_X

$w_{X,t=1} = w_C$
$\phi_{t=1} = \phi$
$R_{X,t=1} = w_{X,t=1} \cdot \phi_{t=1}$
$\pi_{t=1} = valueIteration(R_{X,t=1}, M_X)$
while s_t **not a** terminal state **do**
  Observe s_t
  $\hat{a}_t^{\mathrm{H}} = \pi_t(s_t)$
  Observe a_t^{H}
  if $a_t^{\mathrm{H}} \neq \hat{a}_t^{\mathrm{H}}$ **then**
    if $\Delta p \leq \Delta p_{max}$ **then**
      $w_{prior} = w_{X,t}$
    else
      $\phi' \leftarrow queryUser(\phi_X)$
      $\phi_{t+1} = (\phi_t, \phi')$
      $w_{prior} \sim U(0, 1)$
    Approximate $\hat{\Xi}_{1:T}$
    $w_{X,t+1} = maxEntropyIRL(\hat{\Xi}_{1:T}, w_{prior}, \phi_{t+1})$
    $R_{X,t+1} = w_{X,t+1} \cdot \phi_{t+1}$
    $\pi_{t+1} = valueIteration(R_{X,t+1}, M_X)$

Our system consists of two phases: (i) an offline phase where the robot learns a user preference in the canonical task represented by an MDP, M_C, and transfers it as a prior to the actual task represented by a different MDP, M_X, and (ii) an online phase where the robot predicts user actions at each step of M_X and updates the prior through interaction (Algorithm 1).

3.1 Learning canonical task preferences

To enable the transfer of human preferences, we assume access to a task-agnostic feature function $\phi(s) \in \mathcal{R}^d$ that maps each state s in both the canonical and actual assembly tasks to a d-dimensional feature vector, and model the reward function in both tasks as a linear combination of the features, so that:

$$R_C(s) = w_C^T \phi(s) \; \forall s \in S_C, \quad R_X(s) = w_X^T \phi(s) \; \forall s \in S_X \tag{1}$$

The weights in the d-dimensional weight vector w represent how users value the features ϕ.

Given a set of demonstrated action sequences in the canonical task M_C, we use maximum-entropy IRL [40, 44] to learn the weights w_C. In our implementation, we iteratively update a weight initialized from a uniform distribution to minimize the difference between the expected feature count of the user's preferred sequence and the policy estimated based on the learned weights.

$$\nabla \mathcal{L}(w_C) = \frac{1}{|\Xi_C|} \sum_{\xi_C \in \Xi_C} \sum_{s \in \xi_C} \phi(s) - \sum_{s \in S_C} D_\pi(s) \phi(s) \tag{2}$$

Here, $D_\pi(s)$ is the state visitation frequency for the policy π, computed using the weights w_C, and Ξ_C is the set of user demonstrations $\xi_C = [(s_1, a_1), \ldots, (s_t, a_t), \ldots]$ in the canonical task.

We then transfer the learned preferences by initializing the user's reward function $R_{X,t=1}$ in the actual task with the weights $w_{X,t=1} = w_C$ learned in the canonical task. We provide a simple example of our approach for transferring user preferences from canonical to actual tasks in the supplementary material.

3.2 Updating feature weights

To account for the changing user preferences from the canonical to the actual task, we update the transferred weights based on user corrections in the actual task.

At a time step t, we observe the current state s_t and predict the next human action $\hat{a}_t^{\mathrm{H}}$ based on the user policy $\pi_t(s_t)$, computed with rewards $R_{X,t}$ parameterized by the current estimate $w_{X,t}$ (step 7 in Alg. 1). If the action performed by the user a_t^{H} does not match our prediction $\hat{a}_t^{\mathrm{H}}$, we update the weights as follows.

At the time step t of the actual task, we have only observed the user's action sequence $\xi_{1:t}$ up to that time step. Computing the weights based solely on $\xi_{1:t}$ may be insufficient for inferring the user preference for the rest of the task. To account for the current prior $w_{X,t}$, we update the weights by synthesizing a distribution of action sequences $\hat{\Xi}_{1:T}$ that assumes that the user will execute the previously computed policy π_t for the remainder of the task. We approximate the distribution by sampling trajectories $\hat{\xi}_{1:T} = (\xi_{1:t}, \hat{\xi}_{t+1:T})$. Here, $\xi_{1:t}$ is the observed sequence, $\hat{\xi}_{t+1:T}$ is a sampled sequence computed by executing π_t from the next state s_{t+1}, and T is the total number of time steps until a final state is reached. Similar to the offline setting, we use maximum-entropy IRL [44] to learn the new weights $w_{X,t+1}$ for $\hat{\Xi}_{1:T}$. We iteratively update the weights estimate instead of maintaining a distribution over all weights to enable real-time adaptation during task execution [3, 36].

3.3 Adding new features

In addition to updating the weights of the user's reward function, we consider the case where user preferences in the actual task depend on new features that were not modeled in the canonical task. For example, user preferences in a welding task may depend on the temperature of the assembly. However, if the feature function used in the canonical task does not include a feature for temperature, weights learned over other features will likely not be sufficient for accurately predicting the user's sequence in the actual task.

We assume that in addition to the common set of features ϕ shared between the canonical and actual task, there is a known set of d' different candidate features in the actual task $\phi_X(s) \in \mathcal{R}^{d'}$. We wish to identify which of the candidate features affect user action selection and ask the user to select a feature from the set [7]. We wish to only add features that are relevant to the user preference, since previous work has shown that adding irrelevant features negatively affects performance [11].

To avoid burdening the user, we query them only if there is a big difference between the predicted and the actual user preference. We approximate this difference using as a simple heuristic the number of time steps between the performed action a_t^{H} and the expected time step that the system predicted the user to perform the same action. We only query users if this difference Δp is above a predefined threshold Δp_{max} (step 10 in Alg. 1).

If the user selects a candidate feature ϕ' (step 13), we incorporate it into our feature function with a randomly initialized weight (step 15) and learn the weights for the augmented set of features as in section 3.2. We also remove that feature from the set of candidate features ϕ_X. Otherwise, we update the weights using the existing feature set. Fig. 10 shows an example of the query used in our study.

Table 1: Components included in the proposed system (*online_add*) and in each baseline.

approach	transferred weights	updated weights	feature addition
prior	X		
rand_online		X	
online	X	X	
add_always	X	X	X
online_add	X	X	X

4 SIMULATION EXPERIMENTS

We first evaluate our proposed system with a simulated canonical and actual task to show that - (i) initializing the robot with weights transferred from a canonical task, (ii) updating the weights online, and (iii) incorporating new features that affect user preferences in the actual task contribute to the prediction accuracy.

We design the simulated canonical task (M_C) with $|A_C^{\mathrm{H}}| = 6$ actions and $|S_C| = 27$ states and the actual task (M_X) with $|A_X^{\mathrm{H}}| = 10$ actions, and $|S_X| = 243$ states. We assume a 3-dimensional feature space Φ to model the reward function. We manually design 20 simulated users with substantially different preferences (i.e., weights w), since we found that sampling weights uniformly at random resulted in a small number of distinct preferences.

Benefit of transferred weights. We first compare predicting user actions with online update of the transferred weights (*online*) and randomly initialized weights (*rand_online*). Table 1 shows the differences in each approach.

We consider two scenarios - (1) `same`: Users have the same preference in both the canonical and actual tasks. We use the same weights w to compute the user policy in both tasks. (2) `opposite`: Users have a very different preference in the actual task, compared to the canonical task. We compute the policy in the actual task using different weights $w' = 1 - w$.

In both scenarios, we simulate the users in the canonical task, learn a prior estimate of the users' weights, and calculate the prediction accuracy by comparing the predicted actions $\hat{a}_t^{\mathrm{H}}$ to the simulated actions a_t^{H} in the actual task. The accuracy is 1 when $a_t^{\mathrm{H}} = \hat{a}_t^{\mathrm{H}}$, and 0 otherwise. For each user, we compute the mean accuracy by averaging over all time steps, 25 random seeds and 30 actual task iterations. We use the random seeds to initialize the maximum-entropy learning of the weights w_C in *prior* and *online*, and to uniformly sample the weights $w_{X,t=1}$ in *rand_online*.

Fig. 3 shows that when users have identical preferences in the canonical and actual tasks (`same`), the transferred weights lead to higher accuracy than the random weights. For `same`, a two-tailed paired t-test shows a statistically significant difference ($t(19) = 3.39$, $p = 0.003$) in the accuracy of *online* ($M = 0.73$, $SE = 0.012$) and

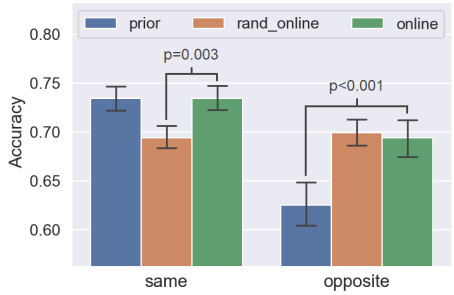

Figure 3: Mean accuracy of predicting user actions in the actual task. The error bars indicate the standard error.

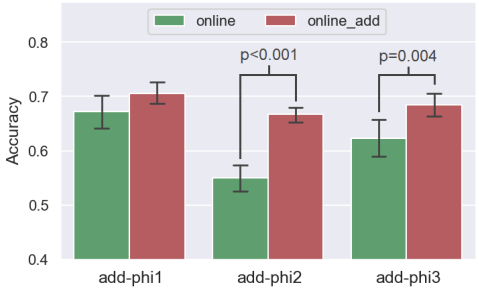

Figure 4: Mean accuracy of predicting user actions using our proposed approach with feature addition for $\Delta p_{max} = 3$.

rand_online ($M = 0.69$, $SE = 0.012$). On the other hand, we do not see a significant difference for users in the `opposite` scenario.

Benefit of updating feature weights. We compare using the transferred weights with (*online*) and without online updates (*prior*). For users that changed their preference (`opposite`), a two-tailed paired t-test shows a statistically significant difference ($t(19) = 5.17$, $p < 0.001$) between the accuracy of *online* ($M = 0.69$, $SE = 0.017$) and *prior* ($M = 0.62$, $SE = 0.023$).

Overall, these results show that when users retain their preference, leveraging the weights learned from the canonical task significantly improves performance, compared to a random prior. If users change their preference, performance is significantly improved by updating the feature weights in the actual task and is comparable to online learning with a uniformly random prior.

Benefit of adding new features. We now consider the case where the preference model learned in the canonical task does not include a feature present in the actual task. We simulate users in the `same` scenario, but when computing their preference in the canonical task, we exclude the first feature ϕ_1 and learn the weights based on the remaining two features (`add-phi1`). We similarly consider two more cases excluding ϕ_2 (`add-phi2`) and ϕ_3 (`add-phi3`).

We compare the proposed approach that uses transferred weights, updates the weights online and adds new features (*online_add*), with using the transferred weights and updating online without adding new features (*online*). We simplify the simulation by adding the excluded feature when $\Delta p > \Delta p_{max}$, without any user selection. In the actual system, the user chooses a relevant feature from a set of candidate features and has the option to not add any feature.

Fig. 4 shows that adding new features (*online_add*) leads to a higher accuracy of action prediction in the actual task than just

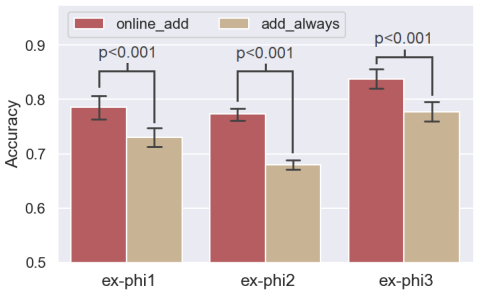

Figure 5: Mean accuracy of predicting user actions using our proposed approach with feature addition based on Δp_{max}.

updating the weights for existing features (*online*). For example, in `add-phi2`, a two-tailed paired t-test shows a statistically significant difference ($t(19) = 4.42$, $p < 0.001$) in the mean accuracy of *online_add* ($M = 0.66$, $SE = 0.02$) and *online* ($M = 0.54$, $SE = 0.02$).

Benefit of selective queries. Finally, we show the importance of adding new features only when $\Delta p > \Delta p_{max}$, by comparing our proposed system (*online_add*) with an identical system that always adds new features, i.e., $\Delta p_{max} = 0$, (*add_always*).

We consider the case where the preference of the simulated users in both canonical and actual tasks depends on only two features, ϕ_2 and ϕ_3, while ϕ_1 is in the candidate feature set (`ex-phi1`). The system *add_always* adds ϕ_1 to the feature set and learns w_1, w_2 and w_3, while *online_add* only considers w_2 and w_3. We do the same for ϕ_2 (`ex-phi2`) and ϕ_3 (`ex-phi3`). Fig. 5 shows that *add_always* performs worse than *online_add*, showing the importance of considering prediction error before adding new features.

Overall, these results show that adding new features based on prediction error improves performance when the features are relevant to the users' preference in the actual task. On the contrary, performance decreases if features are added without considering prediction error.

5 HUMAN-ROBOT ASSEMBLY STUDY

The simulation experiments show the importance of each component of our system. We follow the experiments with a user study, where participants perform an actual assembly task in collaboration with an assistive robot.

The focus of our study is to evaluate the benefit of proactively assisting users using our proposed system. Specifically, we want to show that by anticipating user actions, the robot can reduce the amount of time required to complete the assembly task, improve the team fluency, and affect positively the subjective user experience.

5.1 Study setup

We set up the human-robot assembly task so that at each step of the assembly, the robot fetches the parts required for the next user action from a storage area, and the user performs their preferred assembly action on the workbench (see accompanying video).

We program the robot to ask-before-acting [27], where the user confirms the robot's action so that the robot does not incorrectly deliver the wrong part. We use AprilTags [33] to detect the parts and tools present in the workbench and we recognize user actions using a manually specified dictionary that maps specific part configurations to assembly operations (see supplementary material).

Actual and canonical assembly tasks. We use the same actual and canonical tasks as in our previous work [28]. The actual task is a model-airplane assembly with $|A_X^H| = 8$ actions and $|S_X| = 3324$ states, while the canonical task consists of only $|A_C^H| = 6$ actions and $|S_C| = 175$ states and takes significantly less time for users to complete. We assume deterministic transitions T_C, T_X in both tasks and we set $\Delta p_{max} = 3$ (step 10 in Alg. 1). We provide additional details of both the tasks in the supplementary material.

Feature space. We specify for the canonical task 6 task-agnostic features from our previous work [28] that capture user preferences for selecting actions based on 'physical effort', 'mental effort', 'keeping the same tool' and 'keeping the same part'. Based on a pilot study, we include the 'space required' for actions in the list of candidate features ϕ_X, which captures that in the actual task some users selected actions based on the size of their parts.

5.2 Independent variables

We compare a **proactive (*P*)** robot using the proposed system with a **reactive (*R*)** robot.

When working with a reactive robot R, the user selects the parts required for their next action through a graphical user interface (GUI) and commands the robot to deliver the selected parts. The reactive robot remains stationary in its starting position while the user is performing the assembly and starts moving towards the requested parts only after it receives a command from the user.

In contrast, a proactive robot P uses the proposed system to predict their next action (step 7 in Alg. 1) and proactively reaches to the parts required for that action. We assume a dictionary that associates assembly actions with required parts. In addition to proactively reaching to the required parts, the system displays the predicted action and pre-selects the required parts on the user interface (Fig. 6). If the predicted action is correct, the user can simply confirm the delivery of the pre-selected parts. By reaching to the required parts in advance, P will require less time to deliver the parts to the user. However, if our prediction is inaccurate, the user selects the parts required for their preferred action through the interface. The robot then returns to its starting position before reaching to the correct part, thus requiring more time.

5.3 User study protocol

We recruited 18 ($M = 13$, $F = 5$) participants from the graduate student population of our university, using a sign-up form sent out through the university mailing lists. Participants were compensated with 20 USD. The study protocol was approved by the Institutional Review Board (IRB) at our university.

Participants first provide one demonstration on the canonical task, where they command a reactive robot to deliver desired parts through the GUI. After they complete the canonical task, we compute the preference model for the proactive robot based on the executed action sequence. Participants then execute the actual task with a proactive robot P and a reactive robot R. We counterbalance the order of the P and R conditions to avoid any ordering effects.

We divide both the canonical and actual tasks into a training and an execution phase. In the training phase, participants learn how to perform the task by executing each action in a randomized order. We then ask participants to plan their preferred sequence so that they complete the task in minimum amount of time. We additionally ask them to rate their perceived physical and mental effort for each assembly operation and use their responses to compute the corresponding feature values. In the execution phase, we ask users to perform the actual task according to their planned sequence. After each actual assembly, participants answer a *post-execution questionnaire* (Table 2) and answer open-ended questions about their subjective experience with the robot.

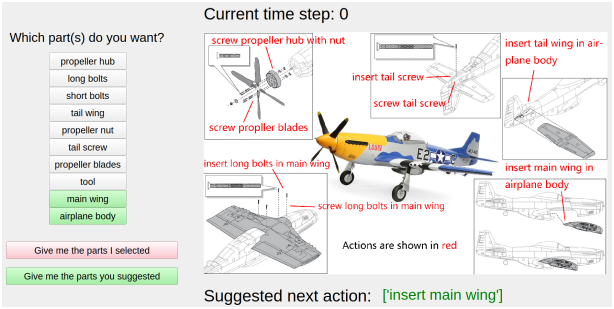

Figure 6: Graphical user interface for interacting with the robot in the actual assembly task. The predicted action and required parts are highlighted in green.

5.4 Hypotheses

We hypothesize that participants will require less time to complete the assembly when working with the proactive robot P, than with the reactive robot R (**H1**). In addition to task efficiency, we expect that proactively assisting users will improve the human-robot team fluency (**H2**), using human idle time as a team fluency metric. We base this on previous work [18, 31] that showed that anticipating user actions significantly improved team fluency.

Next, to show that a proactive robot will have a positive impact on the subjective user experience, we consider the following perceived attributes - team fluency, relative contribution of the robot, user trust in the robot, and robot intelligence. We adopt the scales for fluency, relative contribution, and trust from previous work [17] and we design the remaining questions following recommended practices [37]. We make the following hypotheses: Participants will agree more strongly to statements regarding their perceived fluency (**H3**), relative contribution (**H4**), trust (**H5**), and robot intelligence (**H6**) in the P than the R condition.

5.5 Analysis

Task execution time. We measure the total duration of the task. A two-tailed paired t-test did not show a significant difference in task efficiency between the P ($M = 587.52$, $SE = 24.37$) and R ($M = 593.16$, $SE = 28.116$) conditions, which does not support **H1**. We attribute this result to the large variation in execution times of assembly operations, given that participants were not skilled assembly workers.

Team fluency. We use as team fluency metric the human idle time, which is the time the user spends waiting for the robot to fetch parts. A two-tailed paired t-test showed a statistically significant difference ($t(17) = 5.20$, $p < 0.001$) in the user idle time when working with R ($M = 183.63$, $SE = 2.96$) as compared to P ($M = 174.26$, $SE = 2.98$). In the proactive condition, when the robot

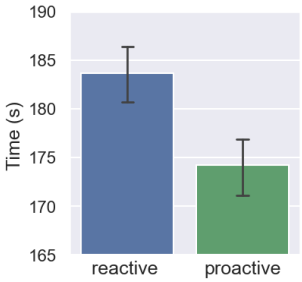

Figure 7: Mean user idle time in the actual task. The error bars indicate the standard error.

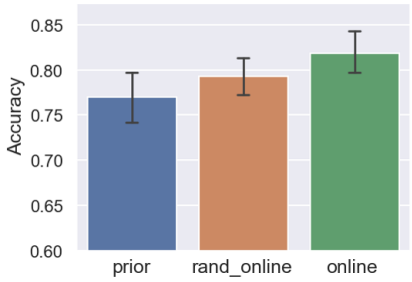

Figure 8: Mean accuracy of predicting user actions in the actual task. The error bars indicate the standard error.

Table 2: Post-execution questionnaire (Likert scales with 7-option response format)

Fluency ($\alpha = 0.94$):
Q1. The robot and I worked fluently together (as a team).
Q2. The robot contributed to the fluency of the interaction.
Relative contribution ($\alpha = 0.83$):
Q3. I had to carry the weight to make the human-robot team better.
Q4. The robot contributed equally to the team performance.
Q5. I was the most important team member on the team.
Q6. The robot was the most important team member on the team.
Trust ($\alpha = 0.84$):
Q7. I trusted the robot to do the right thing at the right time.
Q8. The robot was trustworthy.
Robot intelligence ($\alpha = 0.90$):
Q9. The robot was intelligent.
Q10. The robot does not understand my preferred sequence.
Q11. The robot accurately anticipated my actions.

correctly predicted the participant's next action, idle time decreased because the robot would reach to the required part in advance. On the other hand, if the robot made an inaccurate prediction, it returned to the starting position and the participate had to explicitly annotate the desired parts, resulting in a substantial increase in idle time. Because of the overall high accuracy in the proactive condition (see section 6), the total idle time was decreased. This supports our hypothesis **H2**.

Subjective user experience. We compare the subjective ratings provided by users for the reactive and proactive robots for each scale in the post-execution questionnaire. We measure the internal consistency of each scale by computing the Cronbach's alpha [9] and report it in the questionnaire in Table 2. We treat the combined ratings of all items in each scale as interval data and perform two-tailed paired t-tests [37].

Fluency. We first review the effect of proactive robot assistance on the perceived fluency of the human-robot team using $Q1$ and $Q2$. While users gave a higher mean score for P ($M = 5.72$, $SE = 0.33$) than R ($M = 4.97$, $SE = 0.38$), the test did not show a statistically significant difference ($p = 0.053$). Evaluating $Q2$ only, a Wilcoxon signed-rank test shows a statistically significant difference ($Z = 2.46$, $p = 0.014$) between P ($Mdn = 6.0$) and R ($Mdn = 5.0$). Thus, users recognized that the proactive robot contributed to the team fluency more than the reactive robot. This partially supports **H3**.

Relative contribution. We measure the impact of proactive assistance on the perceived contribution of the robot by combining the scores for $Q3 - Q6$. The test showed a significant difference ($p(17) = 2.52$, $p = 0.021$) in the combined scores for P ($M = 4.2$, $SE = 0.24$) and R ($M = 3.33$, $SE = 0.32$), which supports **H4**.

Trust. A two-tailed paired t-test did not show a significant difference in the user's trust ($Q7 - Q8$) between P and R, which does not support **H5**. We attribute this to the fact that the ask-before-act framework guaranteed that the robot always delivered the correct part in both conditions, thus there were no critical failures that could significantly affect trust in the system [8].

Intelligence. A two-tailed paired t-test showed a statistically significant difference ($t(17) = 4.07$, $p = 0.001$) in the combined scores of $Q9$-$Q11$ for P ($M = 5.44$, $SE = 0.32$) and R ($M = 3.42$, $SE = 0.42$). Thus, **H6** was supported.

User Preference. Overall, 14 out of 18 users reported in open-ended responses that they preferred to work with the proactive robot. They stated that the robot "*predicted what I wanted and moved there*", that the predictions highlighted in the interface required them to perform "*less actions [clicks] for selecting the parts*" and "*reduced their mental load in remembering the next task [action]*".

6 ABLATION STUDIES

We use the data recorded in the assembly study of section 5 to explore how each component of the proposed system contributes to prediction accuracy. We define *prior*, *rand_online* and *online* as specified in Table 1 and evaluate the benefit of initializing the robot with the transferred weights, of updating the weights online, and of adding new features. For each participant of the assembly study, we compute the mean accuracy by averaging over all time steps, 20 random seeds and 5 actual task iterations. We use as ground truth the performed action sequences by that participant.

We first compare the accuracy of initializing our proposed approach with the transferred weights (*online*) to initializing with randomly sampled weights (*rand_online*). A two-tailed paired t-test showed a statistically significant difference ($t(17) = 2.129$, $p = 0.048$) between the accuracy of the *online* ($M = 0.818$, $SE = 0.024$) and the *rand_online* conditions ($M = 0.792$, $SE = 0.019$). We then compare the accuracy of predicting actions with (*online*) and without online updates (*prior*). A two-tailed paired t-test showed a statistically significant difference ($t(17) = 2.707$, $p = 0.015$) in the accuracy between the *online* ($M = 0.818$, $SE = 0.024$) and *prior* conditions ($M = 0.769$, $SE = 0.030$). Thus, we see that *both transferring the preference model from a canonical task and updating the*

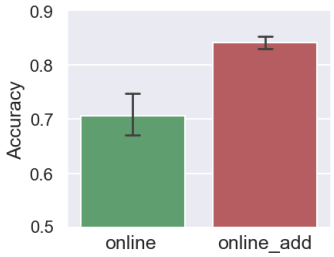

Figure 9: Mean accuracy of predicting user actions using our proposed approach with feature addition (*online_add*) and without (*online*). The error bars are standard errors.

model online are critical for accurately predicting the actions of real users in the actual assembly task.

Lastly, we wish to evaluate the benefit of adding new features. However, in our study only 7 out of 18 participants stated that they considered the additional feature of 'space required' in deciding their preferred sequence of actions and for only 4 out of the 7 participants the feature significantly affected their demonstrated sequence, triggering the condition $\Delta p > \Delta p_{max}$ in line 10, Alg. 1. Because the preferences of most users did not depend on the additional feature in the current setup, we perform a follow-up study to evaluate the benefit of feature addition, as described in section 7.

7 FOLLOW-UP STUDY

To evaluate the accuracy of anticipating user actions using our proposed system with feature addition (*online_add*), we conduct a follow-up study, where we accentuate the importance of a candidate feature that does not appear in the initial preference model. We change our setup of section 5 as follows: (1) We exclude from the feature set ϕ the feature 'keeping the same part' and add it to the list of candidate features ϕ_X. (2) In the study of section 5, participants had to return the current tool to the robot unless their next action required the same tool. Since keeping the same tool often required participants to switch parts, they had to choose between 'keeping the same tool' and 'keeping the same part'. In the follow-up study, we allow participants to retain any used tools, which leads more participants to consider the 'keeping the same part' feature.

We recruited 10 (M=5, F=5) new participants from the graduate student population at our university. For each participant, we followed the same protocol as in section 5, but instead of a reactive robot as baseline, we had a proactive robot that anticipates user actions using our proposed system without feature addition (*online*). A two-tailed paired t-test showed a statistically significant difference ($t(9) = 3.53$, $p = 0.006$) in the accuracy of *online_add* ($M = 0.84$, $SE = 0.012$) and *online* ($M = 0.71$, $SE = 0.04$). Thus, we see that *when user preferences in the actual task depend on features not present in the canonical task, adding the features through user interaction is important for accurate action prediction.*

Fig. 10 shows one of the users in the follow-up study that preferred to 'keep the same part' when sequencing their actions in the actual task. The user starts executing the actual task by consecutively performing actions on the same part, i.e., the airplane propellers. Since the prior transferred from the canonical task does not consider the new feature, the robot incorrectly predicts their action at the second time step. As opposed to the baseline, our

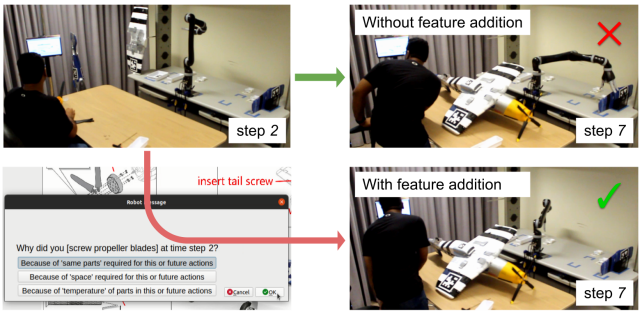

Figure 10: The participant selects the feature for 'keeping same part', when the robot incorrectly predicts their action. By incorporating the selected feature, the robot can later correctly predict the user's action, when the user keeps using the main wing of the airplane. However, if the robot does not add the new feature, the updated weights for the existing features cannot accurately capture the user's preference.

proposed system queries the user with the set of candidate features and the user selects 'keeping the same part'. By incorporating the selected feature and updating the weights, *online_add* is able to accurately predict their action at a future time step 7, where the user prefers to consecutively use the main wing of the airplane.

8 DISCUSSION

Limitations. While we evaluate the proposed system on a single assembly task, future work will test the transfer from canonical to actual tasks in multiple assemblies. Designing the canonical task for a different assembly would require knowledge of the relevant task-agnostic features. For example, in an assembly involving welding operations, we may need to add a feature for 'temperature of parts' and design a canonical task with varying part temperatures.

Furthermore, in addition to asking users to select relevant candidate features with pre-computed feature values, we could ask users to directly assign [7] or teach [6] the values for these features. Moreover, rather than supporting the user in a leader-follower model, the robot could instead reason as an equal partner over the effects of its own actions on the human preference, as part of a mutual adaptation formalism [30]. Finally, while the ask-before-acting approach prevents failures, we expect improvement in team fluency if the robot directly delivers a part when it has high confidence in its prediction, computed with soft-maximum predictive models [45].
Implications. We show the benefit of initializing a preference model with a transferred prior from a canonical task and updating the prior online, to enable robot adaptation to the user preferences in a sequential assembly task. While there has been significant work on preference learning from human inputs directly in the actual task [42], learning from a shorter, expressive task can significantly reduce the burden of time-consuming demonstrations, and we are excited about future work that leverages these advances to learn rich human preference models from canonical tasks as well.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation NRI (# 2024936) and the Alpha Foundation (# AFC820-68).

REFERENCES

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

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

[3] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. 2017. Learning robot objectives from physical human interaction. In *Conference on Robot Learning*. PMLR, 217–226.

[4] Richard Bellman. 1957. A Markovian decision process. *Journal of mathematics and mechanics* 6, 5 (1957), 679–684.

[5] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. 2022. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research* 41, 1 (2022), 45–67.

[6] Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D Dragan. 2021. Feature expansive reward learning: Rethinking human input. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*. 216–224.

[7] Maya Cakmak and Andrea L Thomaz. 2012. Designing robot learners that ask good questions. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 17–24.

[8] Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. 2020. Trust-aware decision making for human-robot collaboration: Model learning and planning. *ACM Transactions on Human-Robot Interaction (THRI)* 9, 2 (2020), 1–23.

[9] Leonard S Feldt, David J Woodruff, and Fathi A Salih. 1987. Statistical inference for coefficient alpha. *Applied psychological measurement* 11, 1 (1987), 93–103.

[10] Lisa R Fournier, Emily Coder, Clark Kogan, Nisha Raghunath, Ezana Taddese, and David A Rosenbaum. 2019. Which task will we choose first? Precrastination and cognitive load in task ordering. *Attention, Perception, & Psychophysics* 81, 2 (2019), 489–503.

[11] Alborz Geramifard, Finale Doshi, Josh Redding, Nicholas Roy, and Jonathan P How. 2011. Online discovery of feature dependencies. In *ICML*.

[12] Matthew Gombolay, Anna Bair, Cindy Huang, and Julie Shah. 2017. Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences. *The International Journal of Robotics Research* 36, 5-7 (2017), 597–617.

[13] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*. 2625–2633.

[14] Elena Corina Grigore, Alessandro Roncone, Olivier Mangin, and Brian Scassellati. 2018. Preference-based assistance prediction for human-robot collaboration tasks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4441–4448.

[15] Yue Guo, Rohit Jena, Dana Hughes, Michael Lewis, and Katia Sycara. 2021. Transfer Learning for Human Navigation and Triage Strategies Prediction in a Simulated Urban Search and Rescue Task. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 784–791.

[16] Kelsey P Hawkins, Shray Bansal, Nam N Vo, and Aaron F Bobick. 2014. Anticipating human actions for collaboration in the presence of task and sensor uncertainty. In *2014 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, 2215–2222.

[17] Guy Hoffman. 2019. Evaluating fluency in human–robot collaboration. *IEEE Transactions on Human-Machine Systems* 49, 3 (2019), 209–218.

[18] Guy Hoffman and Cynthia Breazeal. 2007. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*. 1–8.

[19] Chien-Ming Huang and Bilge Mutlu. 2016. Anticipatory robot control for efficient human-robot collaboration. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 83–90.

[20] Iñaki Iturrate, Luis Montesano, and Javier Minguez. 2010. Robot reinforcement learning using EEG-based reward signals. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4822–4829.

[21] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. 2015. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research* 34, 10 (2015), 1296–1313.

[22] Przemyslaw A Lasota and Julie A Shah. 2015. Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. *Human factors* 57, 1 (2015), 21–33.

[23] Mengxi Li, Alper Canberk, Dylan P Losey, and Dorsa Sadigh. 2021. Learning human objectives from sequences of physical corrections. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2877–2883.

[24] Changchun Liu, Karla Conn, Nilanjan Sarkar, and Wendy Stone. 2008. Online affect detection and robot behavior adaptation for intervention of children with autism. *IEEE Transactions on Robotics* 24, 4 (2008), 883–896.

[25] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. 2017. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*. PMLR, 2285–2294.

[26] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. 2020. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733* (2020).

[27] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. 2018. Efficient behavior learning in human–robot collaboration. *Autonomous Robots* 42, 5 (2018), 1103–1115.

[28] Heramb Nemlekar, Runyu Guan, Guanyang Luo, Satyandra K Gupta, and Stefanos Nikolaidis. 2022. Towards Transferring Human Preferences from Canonical to Actual Assembly Tasks. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 1161–1167.

[29] Heramb Nemlekar, Jignesh Modi, Satyandra K Gupta, and Stefanos Nikolaidis. 2021. Two-Stage Clustering of Human Preferences for Action Prediction in Assembly Tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

[30] Stefanos Nikolaidis, David Hsu, and Siddhartha Srinivasa. 2017. Human-robot mutual adaptation in collaborative tasks: Models and experiments. *The International Journal of Robotics Research* 36, 5-7 (2017), 618–634.

[31] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. 2015. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 189–196.

[32] Stefanos Nikolaidis and Julie Shah. 2013. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.

[33] Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 3400–3407.

[34] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. 2017. Intention-Aware Motion Planning Using Learning Based Human Motion Prediction.. In *Robotics: Science and Systems*.

[35] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. 2020. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020).

[36] Nicholas Rhinehart and Kris M Kitani. 2017. First-person activity forecasting with online inverse reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 3696–3705.

[37] Mariah L Schrum, Michael Johnson, Muyleng Ghuy, and Matthew C Gombolay. 2020. Four years in review: Statistical practices of likert scales in human-robot interaction studies. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. 43–52.

[38] Emmanuel Senft, Paul Baxter, James Kennedy, Séverin Lemaignan, and Tony Belpaeme. 2017. Supervised autonomy for online learning in human-robot interaction. *Pattern Recognition Letters* 99 (2017), 77–86.

[39] Maegan Tucker, Myra Cheng, Ellen Novoseller, Richard Cheng, Yisong Yue, Joel W Burdick, and Aaron D Ames. 2020. Human preference-based learning for high-dimensional optimization of exoskeleton walking gaits. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3423–3430.

[40] Weitian Wang, Rui Li, Yi Chen, Z Max Diekel, and Yunyi Jia. 2018. Facilitating human–robot collaborative tasks by teaching-learning-collaboration from human demonstrations. *IEEE Transactions on Automation Science and Engineering* 16, 2 (2018), 640–653.

[41] Ronald Wilcox, Stefanos Nikolaidis, and Julie Shah. 2013. Optimization of Temporal Dynamics for Adaptive Human-Robot Interaction in Assembly Manufacturing. *Robotics: Science and Systems VIII* (2013), 441.

[42] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. 2017. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research* 18, 136 (2017), 1–46.

[43] Zuyuan Zhu and Huosheng Hu. 2018. Robot learning from demonstration in robotic assembly: A survey. *Robotics* 7, 2 (2018), 17.

[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning.. In *AAAI*, Vol. 8. Chicago, IL, USA, 1433–1438.

[45] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. 2009. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3931–3936.