EFFICIENTLY LEARNING HUMAN PREFERENCES FOR PROACTIVE ROBOT ASSISTANCE IN ASSEMBLY TASKS

by

Heramb Nemlekar

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2023

Dedication

“I dedicate this dissertation to my family and the love of my life.

I will forever be grateful for their unconditional faith, love, and support."

Acknowledgements

I would like to start by thanking my advisor, Stefanos Nikolaidis, for everything that I have learned and accomplished as a member of the ICAROS lab. His insistence on technical rigor, experimental excellence, and clear communication has set the standard for my work. I am deeply grateful for his compassionate leadership during tough times and his dedication to my growth as a researcher. His work ethic has been a constant source of inspiration, and I feel fortunate to have had him as my mentor.

I also want to acknowledge the notable impact of Satyandra K. Gupta on my academic journey. His expertise in presenting research, insights into conference requirements, and feedback during our weekly meetings have been instrumental in shaping my research direction. I value his insights immensely.

I extend my gratitude to the rest of my committee members; Gaurav Sukhatme for his positive feedback on my thesis proposal; Jyotirmoy Deshmukh for pointing me toward RL approaches for online learning; Heather Culbertson for her ideas on conducting longer-duration user studies; and Quan Nguyen for stepping in as the external member at the last moment.

I would also like to thank my collaborators; Neel Dhanaraj for being my partner in weekly meetings, preparing demos, and providing feedback on my research; Barath Raghavan for shaping my research philosophy; Ethan Gordon for teaching me everything about Aikido; and Tapomayukh Bhattacharjee and Amal Nanavati for their crucial words of encouragement after the HRI conference.

I want to express my appreciation for all members of the ICAROS lab who have readily participated in my countless user studies. Matt Fontaine has set the bar high for research quality and output, taught

me everything I know about Quality Diversity, and introduced me to the wonders of Koreatown. Nathan Dennler has always been available to discuss user study protocols. Hejia Zhang's dedication to working at any time has been an inspiration. Bryon, Sophie Hsu, and Varun Bhatt have taken on additional responsibilities and made my life easier. I will cherish the memories of our late-night pushes for deadlines, meme-sharing, and social events.

I am grateful for the opportunity to mentor such talented and motivated students as Ziang Liu, who independently led the robotic lime picking research; Runyu Guan, who helped me in designing the features for transfer learning of human preferences; Guanyang Luo, who conducted numerous user studies; Angelos Guan, who was always there to program robot demonstrations; and Naren Dasan, who ran multiple experiments for testing the transfer of human preferences across various task configurations.

Finally, I would like to thank my friends from neighboring labs, Sandeep, Gautam, Sankalp, Elizabeth, and others, who have made my time in RTH more enjoyable. I also wish to honor the memory of my first friend on campus, Jignesh Modi, who will always be remembered with affection.

Table of Contents

List of Tables

List of Figures

Abstract

Robots that support humans in collaborative tasks need to adapt to the individual preferences of their human partners efficiently. While prior work has mainly focused on learning human preferences from demonstrations in the actual task, obtaining this data can be expensive in real-world settings such as assembly and manufacturing. Thus, this dissertation proposes leveraging prior knowledge of (i) similarities in the preferences of different users in a given task and (ii) similarities in the preferences of a given user in different tasks for efficient robot adaptation. Firstly, to leverage similarities between users, we propose a two-stage approach for clustering user demonstrations to identify the dominant models of user preferences in complex assembly tasks. This allows assistive robots to efficiently infer the preferences of new users by matching their actions to a dominant preference model. We evaluate our approach in an IKEA assembly study and show that it can improve the accuracy of predicting user actions by quickly inferring the user preference. Next, to leverage similarities between tasks, we propose learning user preferences as a function of task-agnostic features (e.g., the mental and physical effort of user actions) from demonstrations in a short canonical task and transferring the preferences to the actual assembly. Obtaining demonstrations in a canonical task requires less time and human effort, allowing robots to learn user preferences efficiently. In a user study with a manually-designed canonical task and an actual task of assembling a model airplane, we observe that our approach can predict user actions in the actual assembly based on the task-agnostic preferences learned in the canonical task. We extend our approach to account for users that change their preferences when switching tasks by updating the transferred user preferences during

the actual task. In a human-robot assembly study, we demonstrate how an assistive robot can adapt to the changing preferences of users and proactively support them, thereby reducing their idle time and enhancing their collaborative experience. Lastly, we propose a method to automatically select a canonical task suitable for the transfer learning of human preferences based on the expressiveness of the task. Our experiments show that transferring user preferences from a short but expressive canonical task improves the accuracy of predicting user actions in longer actual tasks. Overall, this dissertation proposes and evaluates novel approaches for efficiently adapting to human preferences, which can enhance the productivity and satisfaction of human workers in real-world assemblies.

Chapter 1

Introduction

1.1 Motivation

Many assembly tasks, such as spacecraft assembly, are complex and require human dexterity for performing the intricate operations in the task. At the same time, these tasks have a number of simple supporting actions, such as fetching a tool or holding a part in place. Currently, all these supporting actions are performed by humans, leading to low human productivity and high risks to human health.

The advent of human-safe robots has enabled the deployment of human-robot teams in many manual assembly tasks. In these tasks, to enhance human productivity, we would want humans to focus on high-value tasks, e.g., welding or screwing, while robots can perform supporting actions, e.g., bringing parts and tools or clearing out the assembly area. Research in human-human [114, 29, 93] and agent-agent [106] teaming has shown that in high-performing teams, members coordinate their actions by preempting the needs of their partners. Hence, to effectively assist humans, robots need to predict actions that are likely to be performed by their human partners [49, 57, 51, 52, 117]. For example, if the human is expected to perform an assembly operation that requires a screwdriver, the robot can proactively fetch the screwdriver from the tool shelf and deliver it to the human to improve the task efficiency.

There are many assembly, service, repair, installation, and construction applications where the tasks can be performed by several different workers. Even though these tasks have ordering constraints, there

is some variability in the way each worker performs the same task because of their individual preferences [79, 71, 75]. For example, when assembling the spar of an airplane, some workers may prefer to place all the bolts first and then screw them all at once, while other workers may prefer to place and screw one bolt at a time [112]. Thus, to accurately predict human actions, robotic assistants must learn the individual preferences of their human partners. By adapting the supporting actions to the preferences of a human worker, e.g., proactively delivering the bolts in the worker's preferred sequence, the robot can help the worker complete the assembly much more efficiently and fluently [43].

1.2 Goal

Most of the prior research on learning user preferences in executing task actions requires demonstrations of users in the actual task. Typically, this data (e.g., state-action pairs) is used to learn a policy [7, 91] or an underlying reward function [121, 110, 80, 86, 87] that models the preferences of the given user. The learned model can be used to predict the preferred actions of that user in any given state of the actual task. However, obtaining user demonstrations often requires a lot of time and effort, especially in complex and time-consuming tasks like assembly and manufacturing. Consider the setting where each worker performs the actual assembly only once, instead of repeated executions. For example, workers operating in shifts to assemble custom satellites. In such high-mix low-volume settings, obtaining user demonstrations may not be beneficial or even feasible for a task that can only be performed once.

"Our goal is to improve the performance and fluency of the human-robot team in a single execution of an actual assembly task without obtaining user demonstrations on that task beforehand."

1.3 Scope

In this dissertation, we consider the following two scenarios where robots need to assist users with different preferences without having access to their demonstrations in that task:

Scenario 1: A new user is asked to perform the same task that many other users have performed before. We let the robot have access to the demonstrations of the previous users. We want robots to assist the new user according to their preference but without obtaining their demonstration beforehand. In this scenario, we propose that the robot should use the demonstrations of previous users to identify a set of prevalent user preferences and then efficiently infer the preference of the new user during the actual task by matching them to a preference in that set.

Scenario 2: A new user is asked to perform a new task where the robot does not have access to the demonstrations of other users. Without knowing the set of prevalent user preferences in the new task, the robot cannot efficiently infer the preference of the new user. In this scenario, we propose that the robot can instead obtain user demonstrations in a different, much simpler task. We want to enable robots to use their knowledge of the user's preference in the simpler task when assisting them in an actual task that is much more complex.

1.4 Overview

While the space of possible preferences can be very large, previous work has shown that people can be grouped into a few **dominant preferences** [79]. In human-agent teams, users often cluster into a set of "reasonable" behaviors, where people in the same cluster have similar beliefs [88].

Hence, in *Scenario 1*, we propose leveraging similarities in demonstrations of previous users to cluster them into groups of dominant user preferences. Specifically, using insights from clickstream analysis [6], we hierarchically cluster users at different resolutions of the task, i.e., based on the sequence of subtasks

at the high-level and based on the sequence of actions in each subtask at the low-level. Given the set of dominant high- and low-level preferences in a task, the robot uses Bayesian inference to compute the likelihood of a new user belonging to a dominant preference cluster by observing the first few actions performed by the new user in the same task.

Therefore, by leveraging the demonstrations of other users we can enable robots to infer the preference of a new user without having to observe their entire demonstration in the actual task. We evaluate our proposed approach for the first scenario in an IKEA assembly study with 18 participants. Our results show that by identifying the dominant preferences at different task resolutions, we are able to quickly infer the preferences of a new user and predict their actions accurately, when compared to prior work that only clustered users based on their sequence of actions.

However, to accurately infer the preferences of new users, our approach requires access to demonstrations from a sufficiently large population of previous users. This information would not be available to the robot in a new task that has not been performed by several users.

Hence, in *Scenario 2*, we propose learning the preferences of new users by obtaining their demonstrations in a simpler **canonical task** and then transferring the learned preferences to the actual assembly. The canonical task is short so that users can easily provide demonstrations, but also expressive enough so that different users can distinctly convey their individual preferences. Inspired from prior work in the economy of human movement [66, 90, 46] and task ordering [37], our proposed approach models user preference in both the canonical and actual tasks with a common set of *task-agnostic features*, such as the physical and mental effort required by the user to perform the actions in each task. We use the task-agnostic preferences learned in the canonical task to predict the actions of the same user in the new task.

Therefore, by leveraging similarities between tasks based on the task-agnostic features, we can enable robots to predict user actions in a new task using their demonstration in a much simpler task. We evaluate our proposed approach for the second scenario in a user study where 19 participants perform a manually

designed canonical task and an actual task of assembling a model airplane. Our results show that transferring user preferences from a canonical task can enable accurate anticipation of the user actions in an actual assembly when the preferences in both tasks are a function of the same task-agnostic features.

Chapters 2 and 3 describe our proposed approaches for learning the preferences of new users in *Scenario 1* and *Scenario 2*, respectively. In Chapter 4, we extend our approach presented in Chapter 3 to account for users that change their task-agnostic preferences between tasks. Our proposed system updates the transferred preferences during the execution of the actual task by changing the parameters and the set of features in the task-agnostic preference model. We evaluate our proposed approach in a human-robot assembly study with 18 participants where a robotic arm proactively assists users in the model-airplane assembly based on their demonstrations in the manually-designed canonical task. Our results show that by updating the parameters of the transferred preferences during the actual assembly, we can improve the accuracy of predicting user actions and allow the robot to reliably support the user. We also observe that proactively assisting users using our proposed system reduces their idle time and improves their experience of working with the robot, compared to reactive assistance. Further, in a follow-up study with 10 participants, we find that adding new features to the preference model through user interaction allows the robot to assist the users accurately in the remainder of the actual task.

While Chapters 3 and 4 demonstrate the benefit of transferring user preferences from a short canonical task to the actual assembly, they require the canonical task to be manually designed by the experimenter. Hence, in Chapter 5, we propose a method for automatically generating a canonical task that is suitable for the transfer learning of user preferences to any given actual task. Our proposed method selects the canonical task from a set of randomly generated tasks based on their expressiveness, which is the capacity of the task to distinctly learn the preferences of a wide range of users. We evaluate our approach of selecting the canonical task in an online study where 22 participants play an assembly game. Our results

show that preferences learned from user demonstrations in an expressive canonical task achieve a higher accuracy of predicting user actions in the actual task than preferences learned in a non-expressive task.

Chapter 6 summarizes the contributions of this dissertation and discusses future research directions as well as potential applications of our work beyond assembly and manufacturing.

Chapter 2

Learning Dominant User Preferences

2.1 Introduction

We want to enable robots to proactively assist users by predicting their actions in an actual assembly task. The challenge in accurately predicting user actions is that different users can perform the same task in different ways [112]. Thus, to proactively assist a given user, robots must learn their individual preference, i.e., their preferred sequence of actions for completing the assembly task.

While each user can have a slightly different preference and the space of possible preferences can be very large, previous work has shown that people can be grouped into a few **dominant preferences**:

(a) User preference 1 (b) User preference 2

Figure 2.1: Robot-assisted IKEA assembly. (a) Some users preferred to connect all the shelves in a row. (b) Other users connected just two shelves to the small boards first. For effectively assisting the users, a robot must predict their preference and supply the parts accordingly.

People in human-agent teams cluster to a set of 'reasonable' behaviors, and people in the same cluster are likely to have similar beliefs [88]. In game design, players are grouped based on histograms of their actions [89], Likert-scale surveys [104], visits of landmarks [102] or deviations of players' actions to those of a rational agent [50]. Similar clusters exist in education, where students are grouped based on reading times [85], mistakes on exercises [68], and online services [103, 38, 109].

Prior work has shown that such groups of dominant user preferences can be exploited to learn the preferences of unknown users efficiently [79]. Their approach learns the dominant preference models in a simple surface-finishing task by clustering demonstrations of a population of users based on the frequency of their action transitions. After learning the dominant user preferences in the given task, the preferences of new users can be quickly inferred by associating their actions with the closest dominant preference.

However, what makes the problem particularly challenging in complex tasks is that these dominant preferences exist in *different resolutions*. In an IKEA bookcase assembly study, detailed later in this chapter, we observed that some participants preferred to assemble all the shelves in a row, while others alternated between assembling the shelves and assembling the boards (Fig. 2.1). Moreover, within the first group, participants also differed in how they connected the shelves to the boards: some connected all shelves to boards on just one side first (as in Fig. 2.1(a)), while others preferred to connect each shelf to boards on both sides. Thus, at a high-level, users divided the task into sub-tasks of assembling the shelves and the boards, and then considered individual actions in each sub-task at the low-level.

This requires learning the dominant preferences of users at different levels of task abstraction. Using insights from clickstream analysis [6], we propose abstracting the action sequences of users to sequences of *events*, which are the sub-tasks that users divide the task into. The high-level preference of a user is captured by their preferred sequence of events, while the lower-level preference is captured by how they perform the actions in each event. Specifically, we propose hierarchically clustering users on two levels,

over the sequence of events and over the sequence of actions within each event, as opposed to directly clustering based on the sequences of individual actions as in previous work [79].

We show the applicability of our method in the scenario where a new worker executes the same task that we have demonstrations for. If the task has n dominant preferences at all levels, then we need to observe $n \times m$ workers. n is a number usually between three to five [79], while larger values of m allow for more robust inference since there may be variability in the exact sequence of actions of workers with the same preference. We conduct a user study where 20 users assemble an IKEA bookcase, and we show that we can learn the high- and low-level dominant user preferences, which enables accurate prediction for a new user performing the same task. Through an online assembly experiment, we show that assisting users this way improves task efficiency. We finally show the applicability of our approach in a real-world robot-assisted IKEA assembly demo.

2.2 Related Work

2.2.1 Efficiently learning user preferences

User preferences in collaborative tasks are often measured through surveys [32, 24]. The survey responses are used to train a feed-forward neural network that maps the task metrics to the reported user preferences [31]. User preferences for actions that the robot must perform can also be inferred from EEG signals [54], a short window of the human arm motion [20, 96, 84] or the gaze pattern of humans [51]. Past work has also considered user preferences in assigning and scheduling tasks in human-robot teams. Preferences, like the number of tasks that should be assigned to the human, are included as constraints [41, 112] or as part of the objective function [40] in the task scheduling problem.

In task planning, like in our scenario, preference is considered as the subset of action sequences that the human would choose from the set of all sequences that accomplish the given task. Knowing the sequence

of actions that a user will follow in an assembly can enable the robot to preemptively support the user. Robots can learn user preferences from demonstrations provided by the user prior to executing the task [7, 91], or from corrections [3] or feedback [42, 71] provided by the user during the task.

In tasks where humans can visually distinguish their preferred sequences from others, preferences can also be learned by asking users to compare different sequences in the task [94, 14, 15]. The robot asks users to choose their preferred sequence from a set of actively generated sequences. Preference is modeled as a reward function that the user is trying to optimize in the task and the robot's goal is to learn the parameters of the reward function based on the sequence chosen by the user. Since active learning can be computationally expensive when the space of user preferences is large, user demonstrations can be obtained to learn a prior model of user preference [83, 16].

In order to efficiently learn the preferences of a given user, we posit that the robot must have some prior knowledge of the user preferences. Prior work has shown that by identifying the set of *dominant preferences* in a given task, the robot can infer the preference of a new user after observing only a few actions performed by the user in that task [79]. Thus, while each user can have a different preference, our goal is to cluster similar users into a small set of dominant preferences.

2.2.2 Identifying dominant user preferences

Groups of similar users, also called personas, can be built manually from questionnaires [64] or through collaborative filtering [119, 2]. Other related work includes identifying different driver styles [99, 111, 47] using features from vehicle trajectories, human motion prototypes [62] for robot navigation, and human preference stereotypes for human-robot interaction [107].

Most relevant to ours is prior work in identifying dominant user preferences from sequences of user actions in a surface refinishing task [79]. Users with similar action transition matrices were clustered using a hard Expectation Maximization (EM) algorithm to obtain the dominant clusters. However, the task was

simple and thus one-stage clustering of the transition matrices was able to capture the user preferences which were largely encoded in the final position of the robot.

Our key insight is that, in complex tasks, users can have preferences at different resolutions. The problem of grouping users based on their preferred sequence of doing tasks is similar to the problem of *clickstream analysis* [9, 6, 109, 108]. A clickstream is a sequence of timestamped events generated by user actions (clicking or typing) on a webpage and is comparable to the sequence of sub-tasks and actions. Prior work clusters clickstreams of multiple users based on their longest common sub-sequence [9] or frequency of sub-sequences [108].

To cluster users at different resolutions, prior work uses Levenshtein distance to form macro and micro preference clusters [6]. Recent work uses a similarity graph [109] where the similarity between users is measured by comparing the sub-sequences of their clickstreams. Hierarchical clustering is used to partition the similarity graph into high-level clusters which are then further partitioned based on features that were unused for the high-level clustering. We bring these insights from clickstream clustering to the robotics problem of clustering the sequences of users in actual assembly tasks.

2.3 Methodology

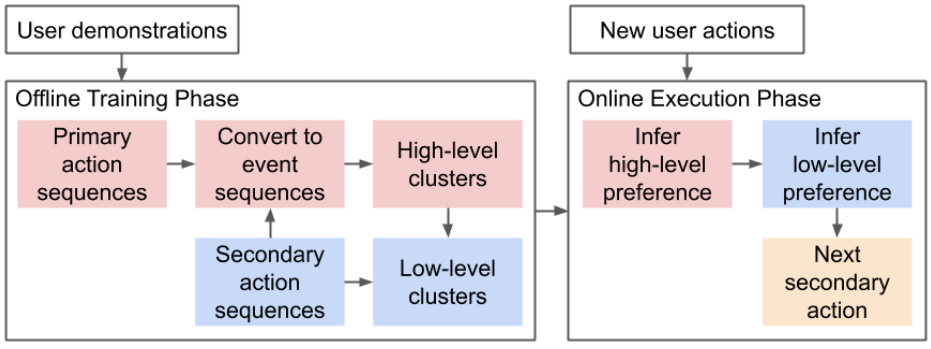


Figure 2.2: Flowchart of our proposed two-stage clustering and inference method.

The proposed method consists of two phases (see Fig. 2.2): (1) an offline training phase which takes as input a set of user demonstrations of the entire assembly task and learns the dominant preference clusters at different resolutions, and (2) an online execution phase where we estimate the probability of a new user belonging to one of the clusters based on their observed actions, and predict the next robot action.

Based on our observation that users prefer to consecutively perform actions that require the same parts, we first convert each user demonstration into a sequence of such $events$. Thus, each event in a demonstration requires a specific set of parts to be supplied by the robot, i.e., a specific set of *secondary actions* (non-critical actions like supplying parts). In complex tasks, there can be several different actions and so the task can comprise multiple events. The *high-level preference* of each user is thus the order in which they perform the events. Further for each event, the users may have a different *low-level preference* for the order in which the set of parts should be supplied, i.e., the order in which the secondary actions must be performed.

We learn the high and low-level preferences in the offline phase by clustering users based on their sequence of events and sequence of secondary actions respectively. Accordingly in the online execution phase, we first infer the high-level preference of a new user and then infer the low-level preference to determine the next secondary action for the robot to execute.

2.3.1 Offline training phase

We assume a set of demonstrated action sequences Ξ, with one demonstration $\xi \in \Xi$ per user. Similar to prior work [43], we distinguish the actions A in the demonstrated sequences into two types: **primary actions** ($a^p \in A^P$) which are the task actions that *must be performed by the user*, and **secondary actions** ($a^s \in A^S$) which are the supporting actions that *can be delegated to the robot.* For instance, a primary action is connecting a shelf, while a secondary action is bringing the shelf to the user. Therefore, $A = \{A^P, A^S\}$.

In the training phase, each user demonstration ξ is some sequence of primary and secondary actions. For example, demonstration $\xi = [a^s_1, a^s_2, a^p_1, a^p_2, \ldots, a^s_M, a^p_N]$ has M secondary and N primary actions. We wish to model the online execution as a turn-taking task with N steps, where at each time step t, a set of secondary actions s_t is performed, followed by a primary action a^p_t. We choose this model as at each time step t, we want the robot to predict the next primary action (a^p_{t+1}) of the user and proactively perform the set of secondary actions (s_{t+1}) that comes before it. Thus, we group consecutive secondary actions into a set of secondary actions s and insert a $NOOP$ (no-operation) action between consecutive primary actions to obtain, $\xi = [s_1, a^p_1, s_2, a^p_2, \ldots, s_N, a^p_N]$. Where in the above example, $s_1 = [a^s_1, a^s_2]$ is the set of secondary actions that must be executed before the primary action a^p_1, while $s_2 = [NOOP]$ means that no other secondary action is required to be executed before a^p_2.

2.3.1.1 Converting demonstrations into event sequences

We first convert each user demonstration to a sequence of *events*. An event e is a sub-task defined as - consecutive primary actions that require the same set of secondary actions.

$$e_{t:t'} := (p_{t:t'}, s_{t:t'}) \quad s.t.\ s_i \subseteq \{s_{t:i-1}\} \quad \forall\ i \in [t+1, \ldots, t'] \tag{2.1}$$

Here, $e_{t:t'}$ is an event from time step t to t', where $p_{t:t'}$ is a sequence of primary actions $[a^p_t, \ldots, a^p_{t'}]$ with preceding secondary actions $s_{t:t'} = [s_t, \ldots, s_{t'}]$ that are similar. Two events are equal if they share the same set of secondary actions - $\{s_t, \ldots, s_{t'}\}$.

Consider the demonstration, $\xi = [[a^s_1, a^s_2], a^p_1, [NOOP], a^p_2, [a^s_3], a^p_3]$. We first set the primary sequence $p_{1:1} = [a^p_1]$ (Line 4 of Algorithm 1) and corresponding secondary sequence $s_{1:1} = [[a^s_1, a^s_2]]$ (Line 10) for event $e_{1:1}$. Then we append the next primary action a^p_2 to the same event if the secondary action preceding it (s_2) is similar to any of the secondary actions in $s_{1:1}$, i.e., $s_2 \subseteq \{s_{1:1}\}$ (Line 6). As $[NOOP]$ is no operation (empty set), we consider it to be a subset of every $\{s_{t:t'}\}$. Therefore, we group $p_{1:2} = [a^p_1, a^p_2]$

and $s_{1:2} = [[a_1^s, a_2^s], [NOOP]]$ into the same event. Similarly, if the secondary action $[a_3^s]$ preceding the next primary action a_3^p is a subset of $\{s_{1:2}\}$, all primary actions belong to the same event $e_{1:3}$, with $p_{1:3} = [a_1^p, a_2^p, a_3^p]$ and $s_{1:3} = [[a_1^s, a_2^s], [NOOP], [a_3^s]]$. In this case, the event sequence is $\xi^e = [e_{1:3}]$. However if $a_3^s \notin [a_1^s, a_2^s]$, then a_3^p will belong to a new event $e_{3:3}$, with $p_{3:3} = [a_3^p]$ and $s_{3:3} = [a_3^s]$. Thus, in this case, the event sequence will be $\xi^e = [e_{1:2}, e_{3:3}]$ where $e_{1:2} = (p_{1:2}, s_{1:2})$.

This process is described in Algorithm 1. We add the primary and secondary actions in a demonstration ξ to an event e (Step 4 and 10) until we reach a new secondary action that is different than the secondary actions in the current event (Step 6). We then append the current event to the event sequence ξ^e (Step 8), and initialize the next event (Step 9) with the new secondary action.

Algorithm 1 Converting user demonstration into sequence of events

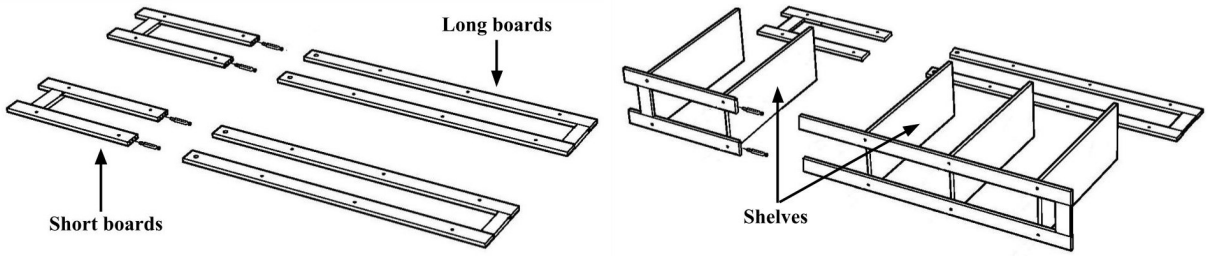
Require: ξ, A^P, A^S
1: **set** $p = [\,]$, $s = [\,]$, $\xi^e = [\,]$
2: **for** a **in** ξ **do**
3: **if** $a \in A^P$ **then**
4: **append** $p \leftarrow [p,\ a]$
5: **if** $a \in A^S$ **then**
6: **if** $s \neq [\,]$ **and** $a \notin s$ **then**
7: **set** $e = (p, s)$
8: **append** $\xi^e \leftarrow [\xi^e,\ e]$
9: **set** $p = [\,]$, $s = [\,]$, $\xi^e = [\,]$
10: **append** $s \leftarrow [s,\ a]$
11: **return** ξ^e

2.3.1.2 Two-stage approach

Clustering event sequences. Once we have converted the user demonstrations Ξ to sequences of events Ξ^e, we cluster the event sequences to determine the *dominant high-level clusters* $z_h \in Z_H$. Details of the method for clustering the sequences are provided in Sec. 2.3.1.3.

Clustering secondary action sequences. To learn the low-level preferences for each event e, we cluster participants based on the sequence of secondary actions in the event to determine the *dominant low-level*

clusters $z_l \in Z_L^e$. We use secondary actions since, from a robot's perspective, primary actions that require the same assistive response (i.e. preceding secondary actions) can be considered identical.



(a) e_{boards} is 1^{st} event for user A

(b) e_{boards} is 2^{nd} event for B and C

Figure 2.3: Visualization of event e_{boards}: connect long and short boards. For B and C, e_{boards} is after event $e_{shelves}$: connect the shelves to the boards.

Intuition. Consider the following example with two events e_{boards} and $e_{shelves}$ (see Fig. 2.3), and three users with event sequences A: $[e_{boards}, e_{shelves}]$, B: $[e_{shelves}, e_{boards}]$, C: $[e_{shelves}, e_{boards}]$. User A prefers to assemble the boards first and then connect shelves to the boards, while users B and C prefer to connect the shelves before assembling the boards. The high-level clustering assigns users B and C to the same cluster, different than A. Here, the set of secondary actions to execute $e_{shelves}$ for users B and C, is different to that for user A. B and C need both the shelf and the board to be supplied as it is the first event, while A only needs the shelves, as the boards were previously supplied for e_{boards}.

Now assume that B performs $e_{shelves}$ by connecting each shelf to the boards on only one side as in Fig. 2.3(b), while C connects each shelf to boards on both sides (not shown). In this case, even though B and C share the same set of secondary actions for event $e_{shelves}$, the sequence of the actions would be different. The robot will perform one $NOOP$ after supplying a shelf to B as it waits for B to connect the shelf to the two boards on one side; whereas for C, the robot will perform three $NOOP$s after supplying the shelf as C connects that shelf to the two boards on both sides. Therefore the low-level clustering will place B and C in different low-level clusters.

In summary, *first stage clustering enables learning the set of secondary actions required, while the second stage specifies the order in which these actions should be performed.* On the other hand, one-stage clustering as in prior work [79] would assign A, B, and C in three different clusters, *losing the information that B and C share the same set of secondary actions*!

2.3.1.3 Clustering method

For each stage, we apply *hierarchical clustering* [11, 69], considering as the distance metric a custom modification (d_{mod}) of the *Levenshtein distance* metric [72] (d_{lev}) used in clickstream analysis [116, 6]. We consider the operators: add and $delete$ as in Levenshtein distance and instead of the $substitute$ operator we use the operation $shift_1$. $delete(i)$ removes the i^{th} element of a sequence. $add(i)$ inserts an element into the i^{th} position of a sequence given that it is empty. $shift_1(i)$ shifts the i^{th} element to the neighboring $i{+}1$ or $i{-}1$ position in the sequence given that it is empty. Each operation has a cost of 1. The $shift_1$ operation allows us to consider sequences that only have two elements in swapped positions as closer than sequences that have two completely different elements in those positions.

Algorithm 2 Greedy approach for calculating $d_{mod}(\xi_a, \xi_b)$

Require: ξ_a, ξ_b

1: $cost_{add} =$ no. of actions in ξ_b that are not in ξ_a
2: $cost_{delete} =$ no. of actions in ξ_a that are not in ξ_b
3: **for** common action in ξ_a and ξ_b **do**
4: $\quad I_a =$ indices of the common action in ξ_a
5: $\quad I_b =$ indices of the common action in ξ_b
6: $\quad C_{I_a,I_b} = \min(shift(i_a, i_b), 2) \quad \forall\ i_a, i_b \in I_a, I_b$
7: $\quad cost_{shift} += greedySearch(C_{I_a,I_b})$
8: **return** $d_{mod} = cost_{add} + cost_{delete} + cost_{shift}$

For two sequences ξ_a and ξ_b, the distance $d_{mod}(\xi_a, \xi_b)$ is the total cost of the minimum number of operations required to transform the sequence ξ_a into ξ_b or vice versa. The number of $add(i)$ and $delete(i)$ operations would be equal to the number of non-common actions between ξ_a and ξ_b (Step 4 and 5 in Algorithm 2). For each common action, we calculate the cost of shifting each instance of the action from

its index i_a in ξ_a to an index i_b in ξ_b (Step 6). We cap the shifting cost at 2 since it is more efficient to perform one add and one $delete$ operation instead for any shift cost > 2. To decide the minimum cost of shifting all instances of a common action, we use a greedy strategy of shifting the instances with the least cost C first (Step 7).

Lastly, the number of clusters depends on a distance threshold. We generate clusters for increasing distance thresholds, and select the optimal distance based on the variance ratio criterion (VRC) [22] (also called *calinski-harabasz score*) which is a common metric for distance-based clustering [30]. VRC is the ratio of the sum of between-cluster dispersion and the sum of inter-cluster dispersion. Therefore for a high VRC the clusters are well separated from each other and the samples within each cluster are dense.

2.3.2 Online execution phase

In the online execution phase, we infer the high- and low-level preferences of new users as they are executing the actual task. At each time step t, as the user performs a primary action, the robot predicts the next secondary action. If the prediction is correct, the robot performs the secondary action and the user can continue with performing the next primary action. If the robot prediction is incorrect, the user performs the desired secondary action according to their preference.

2.3.2.1 Inferring high-level preference

At each time step t, we observe the primary action of a new user and append it to the actions observed so far. We then convert the current sequence of actions $\xi_{1:t}$ of the new user to a sequence of events $\xi^e_{1:t}$ in the same way as in the offline phase. We use Bayesian inference to predict the high-level preference by computing the probability of observing the event sequence $\xi^e_{1:t}$ for each high-level cluster $z_h \in Z_H$.

$$p(z_h|\xi^e_{1:t}) \propto p(\xi^e_{1:t}|z_h)p(z_h) \tag{2.2}$$

Here, $p(z_h)$ is simply the ratio of the number of users in the cluster z_h to the total number of users in all the clusters Z_H. However, for calculating $p(\xi^e_{1:t}|z_h)$, we re-compute the event sequences of users in z_h considering their action sequences only up to the time step t.

$$p(\xi^e_{1:t}|z_h) = \frac{\text{No. of users in } z_h \text{ with same } \xi^e_{1:t}}{\text{Total no. users in } z_h} \tag{2.3}$$

Thus, we can determine the high-level preference as $z^*_h = \arg\max_{z_h \in Z_H} p(z_h|\xi^e_{1:t})$. If there are two high-level clusters with the same maximum probability, we select one randomly.

2.3.2.2 Inferring low-level preference

Once we infer the high-level preference z^*_h of the new user, we identify the most likely event sequence ξ^e in that cluster and assume that the user will follow that sequence for the rest of the task. We then determine the event, e_{t+1}, that the next time step will belong to and the sequence of secondary actions in that event, $s^{e_{t+1}}$, which have been performed by the user so far. We use Bayesian inference to infer the low-level preference $z_l \in Z_L^{e_{t+1}}$, where $Z_L^{e_{t+1}}$ is the set of low-level clusters for the event e_{t+1} in z^*_h.

$$p(z_l|s^{e_{t+1}}) \propto p(s^{e_{t+1}}|z_l)p(z_l) \tag{2.4}$$

We select the most likely low-level preference z^*_l, identically to the high-level preference case. The robot can then perform the most likely secondary action s_{t+1} in z^*_l to proactively assist the user. If the user accepts s_{t+1} we append that to $\xi_{1:t}$. If the user rejects s_{t+1} and performs a different secondary action $s^{'}_{t+1}$, we append $s^{'}_{t+1}$ instead.

2.4 IKEA Assembly Study

We wish to show that the proposed method can effectively identify the dominant preferences of users in an IKEA bookcase assembly task, and use the found preferences to accurately predict the next secondary actions for a new user.

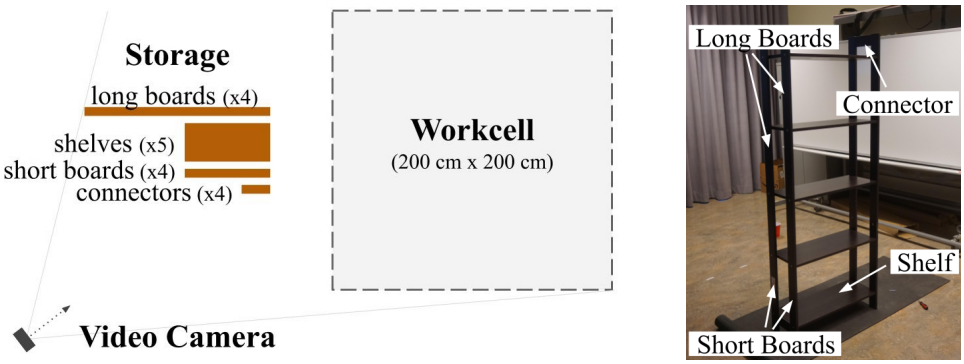


Figure 2.4: Top view of study setup (left) and assembled bookcase (right).

2.4.1 Study setup

We conducted a user study where subjects assembled an IKEA bookcase in a laboratory setting shown in Fig. 2.4-left. We divided the space into a storage area where all the parts were placed and a workcell where the user assembled the bookcase. We recruited 20 participants from the graduate student population at the University of Southern California, out of which 18 ($M = 11$, $F = 7$) successfully completed the study.

2.4.1.1 Procedure

We provided each participant with a labeled image of the bookshelf (Fig. 2.4-right) and demonstrated how the connections are made. Participants then practiced the connections for five minutes. We *did not provide any instructions regarding the order of the assembly.* We informed participants that they had to assemble the bookcase as fast as they can, and asked them to plan their preferred sequence beforehand to ensure

that their plan is well-thought. After the assembly, participants explained their preferences and described how they would want the robot to assist them in open-ended responses.

2.4.1.2 Measurements

We recorded a video of the users assembling the bookcase. We annotated all participants' actions in the video using the video annotation tool ELAN [115]. The annotation was done by 2 independent annotators who followed a common annotation guide.

The study was approved by the Institutional Review Board (IRB) at our university. Participants were compensated with 10 USD and the task lasted about one hour.

2.4.2 Dominant preferences in the bookcase assembly

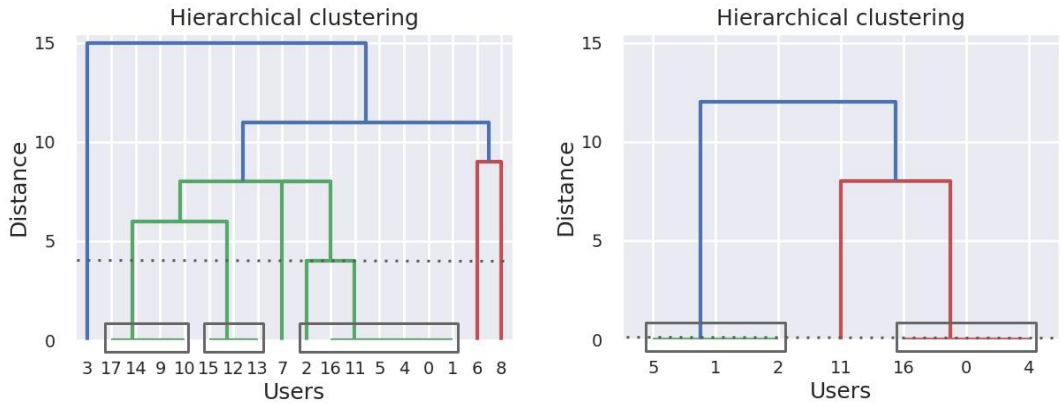
We considered bringing any part from the storage to the workcell as a secondary action and all connections in the assembly as primary actions. The bookcase had 4 types of parts: long boards, short boards, connectors, and shelves (total 17 parts), and 32 different connections. Thus, each user demonstration was a sequence of $N = 32$ time steps.

Figure 2.5: Event sequences in bookcase assembly. 'boards' refers to the event of connecting long and short boards, 'con' refers to the event of connecting boards of the same type using connectors, and 'shelves' refers to the event of connecting shelves to the boards.

2.4.2.1 Event sequences

We first visualize the sequence of events for each user (shown in Fig. 2.5). Users $[0, 1, 4, 5, 11, 16]$ had the same event sequence: short and long board connections (shown in grey), connector and board connections (green), and shelf and board connections (yellow). Similarly, other groups of users like $[12, 13, 15]$, and $[3, 9, 10, 14, 17]$ also had the same event sequences.

The assembly task is fairly complex; the 32 primary actions can be ordered in more than 24! ways! Yet, several users preferred to perform similar actions in a row: 14 users performed all long and short board connections in a row, and 7 users connected all connectors and all shelves in a row.



(a) High-level clusters

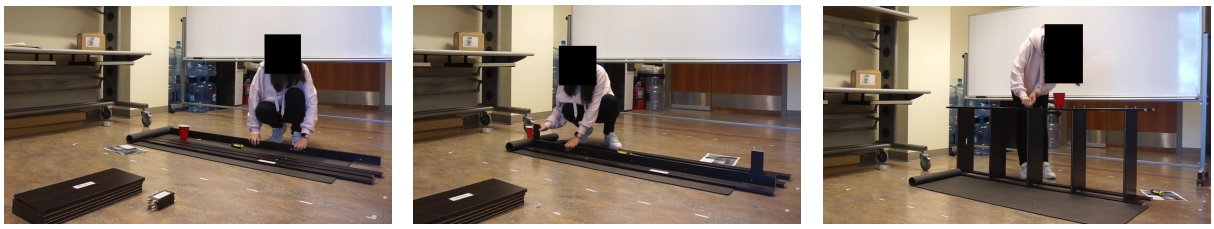
(b) Low-level clusters

Figure 2.6: IKEA assembly user study: (a) Dominant high-level clusters (grey rectangles) formed by clustering the event sequences. (b) Dominant low-level clusters (grey rectangles) formed by clustering the secondary action sequences for the event of all shelf connections (shown in yellow in Fig. 2.5).

2.4.2.2 Dominant clusters

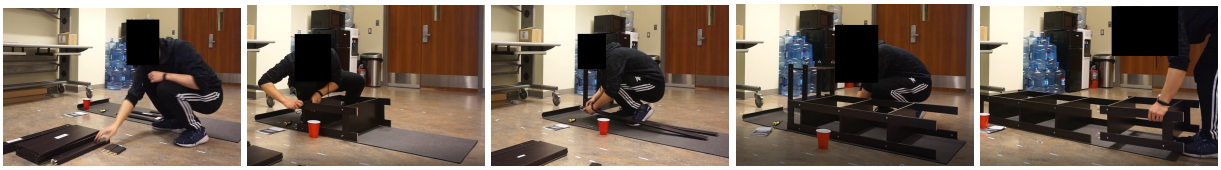
To find the high-level preferences, we cluster the event sequences using the modified Levenshtein distance metric (Section 2.3.1.3) which results in the hierarchy shown in Fig. 2.6(a). We partition the users at a distance threshold of $d_{mod} = 4$ (shown as dotted line) based on the VRC [22] to obtain three dominant high-level clusters (shown in grey rectangles). Therefore, we see that in a relatively complex bookcase

assembly, users cluster into just three dominant high-level preferences (see Fig. 2.7, 2.8, and 2.9), despite not being provided any instructions regarding the order of the connections.



(a) boards (b) connectors (c) shelves

Figure 2.7: Dominant high-level preference 1: (a) Users first connect all the long and short *boards* in a row, (b) then assemble the boards with *connectors*, and (c) finally screw all *shelves* to the boards.



(a) connectors (b) shelves (c) connectors (d) shelves (e) boards

Figure 2.8: Dominant high-level preference 2: (a) Users first assemble the short boards with *connectors*, (b) then screw two *shelves* to the short boards, (c) then assemble the long boards with *connectors*, (d) screw the remaining *shelves* to the long boards, and (e) finally connect all the long and short *boards*.

(a) connectors (b) shelves (c) boards (d) connectors (e) shelves

Figure 2.9: Dominant high-level preference 3: (a) Users first assemble the short boards with *connectors*, (b) then screw two *shelves* to the short boards, (c) then connect the long *boards* to the short, (d) assemble the long boards with *connectors*, and (e) finally screw the remaining *shelves* to the long boards.

Users had different preferences for the same event as well. For example, within the event of performing all shelf connections in a row, shown in Fig. 2.7(c), clustering the sequences of secondary actions of all users results in the hierarchy shown in Fig. 2.6(b). Partitioning at the optimal distance threshold $d_{mod} = 0$ gives us two dominant low-level clusters (shown in grey rectangles). Users 0, 4, and 16 prefer to connect each

shelf to boards on one side before connecting on the other side, whereas users 1, 2, and 5 prefer to connect each shelf to boards on both sides at a time (see Fig. 2.10).

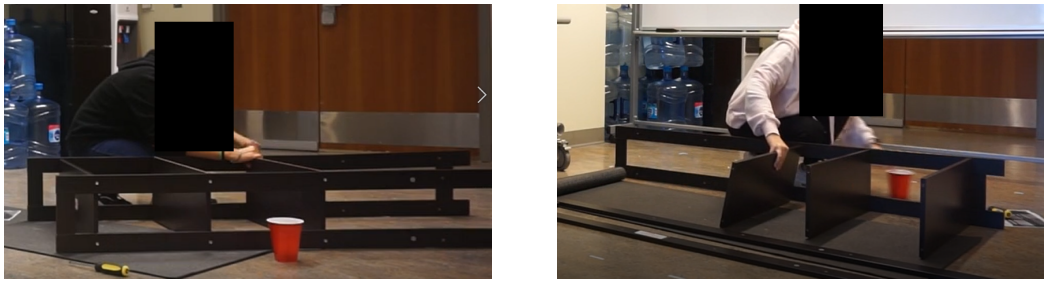


Figure 2.10: Dominant low-level preferences: (Left) Users connect each shelf to the boards on both sides and (Right) users connect each shelf to the boards on one side first.

2.4.3 Open-ended responses

At the end of the study, we asked users to specify the kind of support they would like a robot to provide for efficiently completing the task. Most of the users expressed that they would like the robot to assist them by *"screwing, holding the shelf, keeping the shelf stable, passing me the objects"*. In our experiments, we only consider the action of passing objects to the user as a secondary action that the robot can perform.

Users also commented that the robot should support them by *"giving the parts in an order"*, *"screwing in the screws after I put them in"*, and *"securing a part while I attach something"*. This would require the robot to predict and proactively perform the next secondary action, e.g., the robot can fetch the screwdriver while the user is putting in the screws.

2.4.4 Experimental evaluation

We want to show that our two-stage clustering approach is better at predicting the next secondary action of the robot for a new user, as compared to a one-stage approach. We make the following hypothesis:

H1. Clustering users based just on their sequence of *events* improves the accuracy of predicting the next secondary action, compared to clustering based on the sequences of individual primary actions.

H2. The two-stage clustering of users improves the accuracy of predicting the next secondary action, compared to clustering based just on the sequences of events.

2.4.4.1 Experiment details

Using the demonstrated action sequences in the user study, we perform leave-one-out cross-validation, where we exclude each participant one by one and generate the preference clusters from demonstrations of the rest of the participants. For each time step in the demonstration of the excluded user, we hierarchically infer the high- and low-level preference of the participant based on the sequence of actions performed until that step (Sec. 2.3.2) and predict the next secondary action as per the proposed approach. In the end, we compare the cross-validation accuracy of predicting the next secondary action at each time step, averaged over 100 trials for each excluded user.

(a) Importance of high-level

(b) Importance of low-level

Figure 2.11: Cross-validation accuracy for 18 users averaged over 100 trials. (a) Clustering event sequences compared to clustering primary action sequences. (b) Two-stage clustering compared to clustering event sequences.

2.4.4.2 Importance of high-level clusters

First, we compare the prediction results from clustering based on sequence of events (without second-stage clustering) to clustering based on the sequence of primary actions. While clustering based on event sequences leads to three dominant clusters, clustering based on primary action sequences generates only

two dominant clusters for a distance threshold of 63 (based on VRC), resulting in high variability in the sequences in each cluster. A two-tailed paired t-test showed a statistically significant difference ($t(17) = -3.232,\ p = 0.004$) in prediction accuracy averaged over all timesteps and trials, between clustering based on event sequences ($M = 0.796,\ SE = 0.041$) and clustering based on action sequences ($M = 0.693,\ SE = 0.041$). This result supports hypothesis **H1**. Thus, in assemblies that can be divided into subtasks, we see that users can be better grouped together based on their sequence of events to identify the dominant user preferences at the high-level.

2.4.4.3 Importance of low-level clusters

We also compare the prediction accuracy with and without the second stage clustering, averaged over all timesteps and trials. A two-tailed paired t-test showed a statistically significant difference ($t(17) = -2.34,\ p = 0.03$) in the prediction accuracy of clustering based only on the event sequences ($M = 0.796,\ SE = 0.041$) and the proposed two-stage approach ($M = 0.820,\ SE = 0.043$). This result supports hypothesis **H2**. Thus, we see that users can also have dominant preferences for how they execute the actions in each subtask at the low-level.

2.4.4.4 Interpretation of results

We observe that clustering based on event sequences outperforms clustering based on primary actions. The baseline performs poorly after timestep $t = 12$ (Fig. 2.11(a)) as most users switch to shelf connections; there, the order of primary actions (which shelf to connect) is different among users, but they require the same secondary actions (bringing a shelf) and thus belong to the same event. The drops in accuracy for our method are caused by errors in prediction for participants that did not belong to any cluster, e.g., users 3, 6, 7, and 8 in Fig. 2.5, as well as by "branching" points, where participants that had identical sequences

started to differentiate. For instance, participants 14 and 15 differentiate at $t = 12$, where performance drops for all methods.

Similarly, we observe in Fig. 2.11(b) that the two-stage clustering and clustering based on events only perform similarly up to $t = 18$, from where users differentiate on how they connect the shelves to the boards, which is where the low-level preferences benefit prediction.

2.5 Online Assembly Experiment

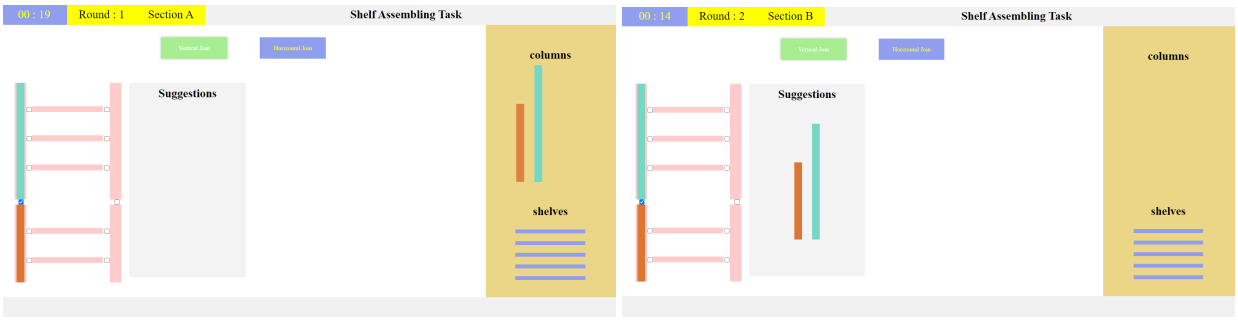


Figure 2.12: Online shelf assembly game. (Left) Without assistance, users have to get the parts for their next action from the storage at the right of the screen. (Right) With assistance, the robot delivers the parts for the next user action close to the assembly.

We wish to show that predicting the secondary action to assist a new user improves the efficiency of the task. Therefore, we implemented an online shelf assembly game (see Fig. 2.12). In this game, users have to click and drag parts placed on the right side of the screen to the assembly workcell on the left. Once they place the parts of their next action, users must select the correct tool button and then click the checkbox between the two parts to connect them.

We first conducted an online study where 20 users demonstrated their preference for playing the game and used that to learn the dominant high- and low-level preferences. As the game was simple, users had one dominant high-level preference and two low-level preferences on how to connect the shelves, which interestingly matched closely with how users clustered when connecting shelves in the real world (Fig. 2.6(b)). We then conducted an online experiment with 80 participants recruited through Amazon

Mechanical Turk, where users played a shelf assembly game three times each with and without assistance. When playing with assistance, the system predicts the next secondary action of the user and places the corresponding parts in the suggestions box near the assembly. If the prediction is correct, the user can quickly use the parts without having to get them from the storage. The order of assistance versus no assistance was counterbalanced to reduce any learning effects.

H3. We hypothesize that providing assistance to users by predicting the next secondary action improves the efficiency of the task, as compared to providing no assistance.

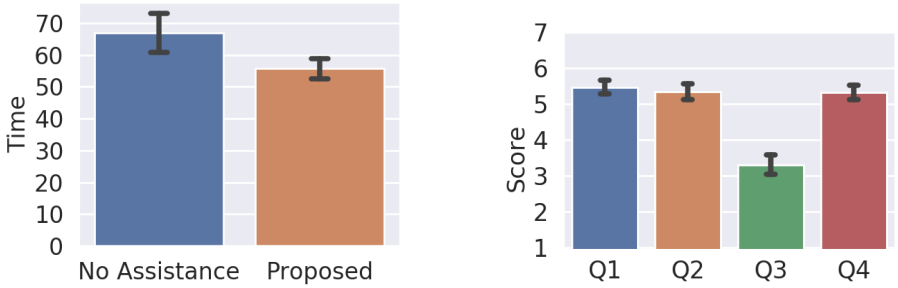
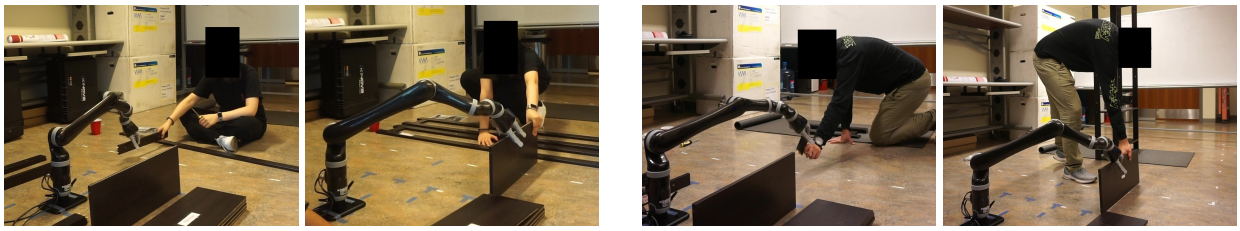


Figure 2.13: (Left) Average time taken by users to complete the online game with and without assistance. (Right) Subjective responses of users for Q1: Assistance was according to your preference?, Q2: Assistance was helpful?, Q3: Assistance was detrimental?, and Q4: Assistance reduced your effort?

We compare the average time taken by 52 (out of 80) users who completed all game trials and survey questions when playing with and without assistance using our proposed two-stage approach (see Fig. 2.13). A two-tailed paired t-test showed a statistically significant difference ($t(51) = 2.155$, $p = 0.036$) in the time required to assemble with ($M = 55.794,\ SE = 3.439$) without assistance ($M = 66.948,\ SE = 6.05$). Moreover, in the subjective responses, the users gave a high rating when asked if the assistance was according to their preference (Q1), was helpful (Q2), and reduced their effort (Q4). Accordingly, they gave a low rating when asked if the assistance made their task more difficult (Q3). This informs us that performing the secondary actions as per our proposed method can indeed be helpful for the users.



(a) User 1 (b) User 2

Figure 2.14: Robot predicts and performs the secondary actions of delivering parts to the users. The users stay in the workcell and perform the primary actions of connecting the parts.

2.6 Robot-Assisted IKEA Assembly

Lastly we show how our proposed method can be used in a human-robot collaborative assembly setting. We demonstrate the online execution phase in the IKEA bookcase assembly task with two participants*, who were instructed to perform the task in two different ways. We used the same setup as shown in Fig. 2.4 with a Kinova Gen 2 robot arm in the storage area.

The human-robot collaboration followed a turn-taking model where the user and the robot alternated in performing one primary action and one set of secondary actions, respectively. Therefore, the participant always stayed inside the workcell and performed all the connections, while the robot brought the required parts from the storage area to the workcell (Fig. 2.14). The experimenter inputted each primary action performed by the participant into the system, and the system inferred the users' preferences and selected the next secondary action for the robot to perform. We pre-planned the motion of the robot arm to reach the part that needs to be delivered, grasp it, and move it to the workcell. We saved these trajectories in a look-up table to make the robot response faster.

Videos of the robot demonstration and the online assembly experiment can be found at youtu.be/2yL89vlhG74 and youtu.be/wDHQXBua4OI. We observe that the robot assistance allows users to remain inside the workcell and focus on the value-added work. We hypothesize that this can be beneficial, not just in

*We were unable to perform a complete human subjects experiment because of COVID-19 restrictions.

improving the efficiency of the task but also in reducing the human effort, and we plan to explore this in future work.

2.7 Conclusion

We propose a two-stage clustering approach, inspired by clickstream analysis techniques, to identify the dominant preferences of users at different resolutions in a complex IKEA assembly task. We show how this approach can enable prediction of the next action required of the robotic assistant and how it can improve the efficiency of task execution.

A limitation of our work is that we do not model the effect of robot actions on the user [95, 78], or the utility of actions with respect to team performance. Future work can also consider the confidence in the prediction, to decide if a robot should directly perform the predicted action or request additional information from the user to prevent failures.

While we have focused on predicting the actions of a new worker on the same task that we have demonstrations for, our insights from this study can be applied to the problem of having the same worker perform a new, slightly different task. Even though the order of events may be different, and some events may be task-specific, we can learn user preferences for any task-agnostic aspects of the events that are shared among the two tasks to proactively assist the user. The next chapter explores how preferences of the same user can transferred from one assembly task to another for predicting user actions.

Chapter 3

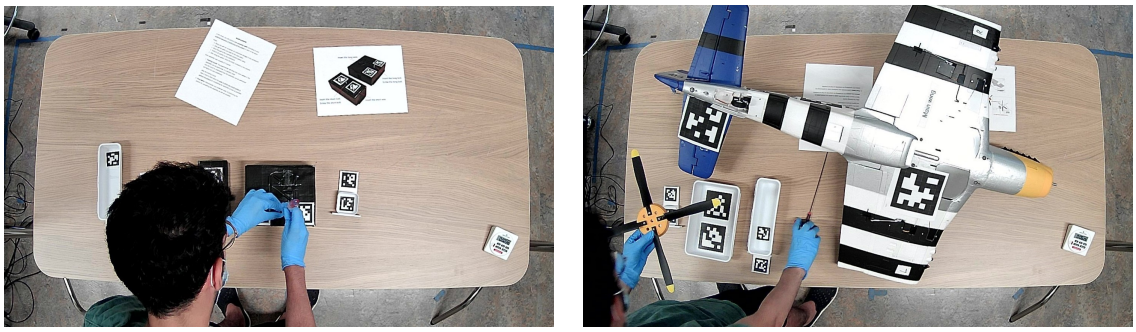
Transferring Human Preferences from Canonical Tasks

3.1 Introduction

In the previous chapter, we considered the scenario where a new user is asked to perform the same task that several users have performed before. In that scenario, the robot can learn the dominant user preferences in the actual task by clustering the demonstrations of previous users and then infer the preference of a new user by associating them to a dominant preference model [79, 75]. However, to accurately model the dominant user preferences and infer the preferences of different users in a complex task, the robot would need access to a large number of previous demonstrations. Moreover, if the assembly task changes, the dominant preferences learned in the previous task may no longer be useful in the new task.

In this chapter, we consider the scenario where the user is asked to perform a completely new task. Specifically, we consider that the task is a customized assembly where the robot must assist each user in only a single execution of the task. Here, the robot does not have access to any prior demonstrations in the actual assembly to learn the dominant user preferences. Moreover, since the assembly will only be performed once by a given user, we do not wish to obtain demonstrations from the user in the actual task in order to reduce the human effort and maximize productivity.

"How can we efficiently learn the preferences of new users to accurately predict their actions without obtaining any demonstrations in the actual assembly task?"



(a) Canonical assembly task (b) Actual assembly task

Figure 3.1: Example of a user that prefers to perform high-effort actions at the end of assembly tasks. (a) Last action of the user in the canonical task is to screw the long bolt which requires the most physical effort. (b) Second last action of user in the actual task is to screw the intricate propeller which also requires the most physical effort (as rated by the user).

In the bookcase assembly study in Section 2.4, users explained their preferences in terms of features like the physical or mental effort they required. Some users preferred to perform all the similar actions in a row to reduce the mental effort required to switch from one type of action to another, while other users preferred to assemble the bookcase vertically to reduce their physical effort even if they had to switch between dissimilar actions. Inspired from prior work in economy of human movement [66, 118, 90, 46] and task ordering [37], our key insight is that user preference across different yet related assembly tasks can be represented with a common set of *task-agnostic features*, such as the movement and physical and mental effort required by the user to perform the assembly operations.

Thus, instead of learning from demonstrations in the actual task, we propose learning the task-agnostic preferences of a new user from their demonstrations in a much shorter, **canonical task** and transferring the preferences to the actual task to predict user actions. We wish our canonical task to be short so that users can easily provide demonstrations, but also expressive enough so that different users can clearly demonstrate their task-agnostic preferences. For a given user, we hypothesize that their preferences over the task-agnostic features will be similar in both the canonical and actual tasks. For example, if a worker prefers to perform the high-effort actions at the end of the canonical task, they will likely prefer the same in the actual task (see Fig. 4.9).

In this work, we empirically design a canonical task for a given assembly, and focus on investigating whether human preferences can be effectively transferred from canonical to actual assembly tasks. Our problem is especially challenging and distinct from prior work in transfer learning [100, 19, 27], since we focus on transferring preferences of real users – as opposed to agent policies – across tasks.

We assume that, in addition to the task objectives, users perform the assembly with the intention of optimizing some combination of their physical and mental efforts. Thus, we model the internal objectives of users as a linear function of the task-agnostic features, where the feature weights represent the individual user preference. We use the maximum-entropy inverse reinforcement learning (IRL) [121] approach to learn the weights for each user from their demonstration in a canonical task and then use the same weights to model their rewards and compute their policy in the actual task.

In this chapter, we show how the preferences of real users can be transferred from a canonical to an actual assembly task. We validate our proposed method in a user study where 19 participants demonstrate their preferred way of performing a short canonical task and an actual task of assembling a model-airplane. Our results show that by transferring user preferences from the canonical task we can predict user actions in the actual assembly with significantly higher accuracy than random action predictions.

3.2 Related Work

Similar to prior work [79, 94, 83], we consider that the preferences of a user are captured by their internal reward function. Therefore, our goal is to transfer the user's reward function from a canonical task to an actual assembly.

3.2.1 Transferring human preference from source to target task

Prior work in *transfer learning* [101, 19, 27] has explored transferring agent policies from a source task to a target task by assuming a mapping between the states and actions of the two tasks. For example, the $left$

action in a 2D mountain car task is mapped to $left$ and up in a 3D version of the same task [19]. Similarly, the $right$ action in the 2D version is mapped to $right$ and $down$ in 3D. In another example, the rewards for states in a 3-vs-2 robot soccer task are simply repeated for the extra states in a 4-vs-3 task [100]. However, finding an intuitive mapping between different assembly tasks is not as trivial, especially for transferring human preferences.

The problem of transferring the preferences of real users is distinct from the problem of transfer learning that focuses on using the policies of robotic or simulated agents in a source task as priors to speed up learning in the target task. Instead, our work focuses on learning the user's preference as a function of task-agnostic features for anticipating their actions in a different assembly task. Moreover, unlike prior work in transfer learning [100, 28, 56, 23], we attempt to transfer user preferences without access to the user's rewards or demonstrations in the target task.

The prior work most similar to our problem transfers the preference of a simulated human from a (source) block stacking task to a target task with an extra red block [71] by modeling artificial preferences that are a function of the block color. However, to our knowledge, no prior work has studied transferring preferences of real users from one assembly task to another.

3.2.2 Factors affecting human preferences in physical tasks

User preferences for sequencing the actions (or sub-tasks) in assembly and manufacturing tasks can be affected by several factors that are task-specific. For example, in a part stacking assembly [110], user preferences depended on the size and color of the parts, and were modelled as a linear reward function of the two features. While in a Lego model assembly [41], user preferences for task allocation depended on the type of sub-task - fetching or building.

In order to transfer user preferences from one assembly task to another, we want to model the preferences based on features that are *task-agnostic*. Previous studies [66, 90, 46] have shown that users prefer to

minimize movements during task execution. We expect the same for users in an assembly task, i.e., users would prefer to minimize movements required to perform the assembly. Similarly, users may also look to minimize their effort spent in the task. For example, in an object pick-up task [37] some users preferred to pick up the closest object first because it reduced their cognitive effort, even if it increased their physical effort in the long run. In a study on human jump landing [118], users optimized a combination of active and passive efforts, while also accounting for other factors like safety.

In this work, we presume that users will prefer to minimize some combination of the movement cost and the physical and mental efforts, with different users having different combinations (i.e. preferences).

3.3 Methodology

We want to transfer user preferences from a canonical task M_C to an actual assembly task M_X for anticipating user actions. We model each task as a Markov Decision Process (MDP) defined by the tuple $M := (S, A, T, R)$, where S is the set of states in the assembly, A is the set of actions that must be performed to complete the assembly, $T(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state $s_{t+1} \in S$ from state $s_t \in S$ by taking action $a_t \in A$, and $R(s_{t+1})$ is the reward received by the user in s_{t+1}.

We assume that S_X, A_X, and T_X are known for the actual assembly task M_X. While T_X models the ordering constraints, since each worker can have their own preferred sequence, $\xi_X = [a_1, \ldots, a_t, \ldots, a_N]$, of performing the actions $a_t \in A_X$, R_X will be specific to each worker. Therefore, to anticipate the actions of a user i in the actual assembly task, we must learn their individual reward function $R_{X,i}$.

Goal. We want to learn the user's reward function R_X from demonstrations ξ_C provided by the user in a canonical task M_C (which has its own set of S_C, A_C and T_C).

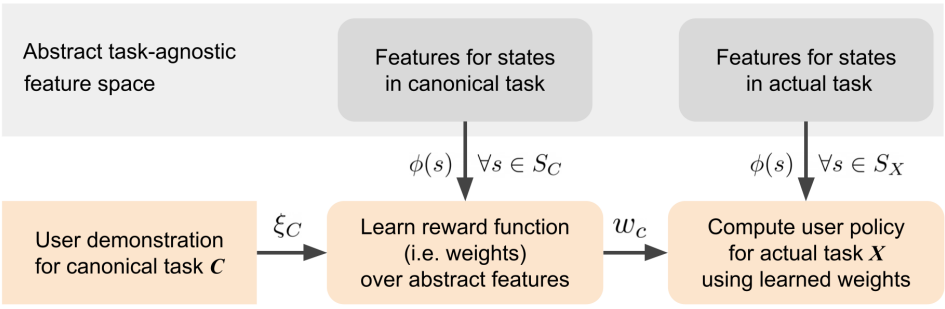


Figure 3.2: Flowchart of our proposed method for transferring user preferences from a canonical task to the actual assembly task.

3.3.1 Intuition

Our key insight is that preferences of users in actual assembly tasks can be represented with abstract features from a task-agnostic feature space Φ. Given a feature function $\phi : \{S_C, S_X\} \mapsto \Phi$, we can map any state in the canonical and actual tasks to a d-dimensional feature vector in Φ.

As the rewards received by a user depend on the state of the task, using ϕ, we can model the reward function of a given user i as a function of the task-agnostic features.

$$R_{X,i}(s) = f_{X,i}(\phi(s)) \quad \forall\ s \in S_X$$

The function $f_{X,i}$ is specific to the user i, and captures their individual preference. Our hypothesis is that users will have similar preferences over the abstract features in both the canonical and actual tasks, i.e., $f_{X,i} \simeq f_{C,i}$, given that: (i) the feature space Φ fully captures the preferences of all users, and (ii) the canonical task M_C is expressive enough to capture preferences over a diverse range of feature values.

Given the task-agnostic feature function ϕ, we can learn the user's $f_{C,i}$ from their demonstrations ξ_C in the canonical task, and use the same function, i.e., $f_{X,i} = f_{C,i}$, to calculate their rewards $R_{X,i}$ for the states in the actual assembly task.

3.3.2 Approach for transferring user preferences

Following prior work [121], we model the reward function $R_{X,i}(s) = w_{X,i}^T \phi(s)$ as linear in the features $\phi(s)$. Here, w is a d-dimensional weight vector where each weight in the vector represents the user's preference for a particular dimension of the feature space.

Given a demonstration sequence $\xi_C = [a_1, \ldots, a_N]$ of actions $a \in A_C$, we use maximum-entropy IRL [121] to learn the weights w_C for the user. In this approach, we iteratively update the weights to maximize entropy (as there can be multiple solutions) such that our learner visits the features in the canonical task with the same frequency as observed in ξ_C. We choose the maximum entropy approach because we wish the learned weights to explain the demonstrations without adding any additional constraints to the resulting policy.

Based on our key insight, we assume that user i would have the same weights ($w_{X,i} \simeq w_{C,i}$) for the features ϕ in the actual task. Thus, we can calculate the transferred rewards R_X by using the weights w_C:

$$R_{X,i}(s) \simeq w_{C,i}^T \phi(s) \quad \forall s \in S_X \tag{3.1}$$

Fig. 3.2 summarizes our proposed approach for transferring preferences from canonical to actual tasks. We conduct a user study to evaluate whether R_X can be used to effectively anticipate user actions in the actual task.

To anticipate user actions, we assume that the user will try to maximize their long-term reward in the actual assembly. Thus, we use the learned R_X to perform value iteration [12] for all states $s \in S_X$ in the actual task, and select the action with the highest value as our prediction $\hat{a}_t$ in a given state. In our study, we calculate the value of taking an action in a given state without discounting the future rewards, since users plan for the entire assembly before they demonstrate their preference.*

* For very long assembly tasks users might minimize movement or effort over a shorter horizon, in which case we would adapt the time horizon accordingly.

3.4 Task-Agnostic Feature Space

Inspired from prior work in economy of human movement [66, 90, 46], we presume that users would try to minimize their movement throughout the task. Because the set of actions in our assembly tasks is fixed, users can minimize their movement when they switch from one action to the next. For example, a worker may prefer to consecutively perform all the actions on one side of the assembly to avoid having to shift sides. In our user study, movement for switching between actions is performed when the user changes the tool or the part they are working on. Thus, we consider the following features to capture user preferences for minimizing movement:

Table 3.1: Movement-based features.

Feature	Weight	Value	Preference
$\phi_{\mathcal{P}}$	$w_{\mathcal{P}}$	$\{0, 1\}$	Keep same part
$\phi_{\mathcal{T}}$	$w_{\mathcal{T}}$	$\{0, 1\}$	Keep same tool

Here, $w_{\mathcal{P}}$ and $w_{\mathcal{T}}$ are the weights for the respective movement-based features in the user's reward function. For a state s_{t+1}, $\phi_{\mathcal{P}}(s_{t+1}) = 1$ if the latest action a_t uses the same part as the previous action a_{t-1}. Similarly, $\phi_{\mathcal{T}}(s_{t+1}) = 1$ when a_t requires the same tool as a_{t-1}. Because we want our features to be state dependent, we augment the current state with the previous two actions: $s_{t+1} \leftarrow [s_{t+1}, a_t, a_{t-1}]$. At the start of the task, the previous two actions in the start state are set to $None$.

Previous work [118] also states that users may choose actions trying to minimize the effort they spend in the task. For each action, we define the effort ε as a weighted combination of the nominal physical (ε_p) and mental (ε_m) efforts required to perform that action.

$$\varepsilon(a) = w_p \varepsilon_p(a) + w_m \varepsilon_m(a) \tag{3.2}$$

Here, the weights w_p and w_m are specific to the user and depend on their individual preference towards minimizing their physical and mental effort. For example, users may prefer to perform specific actions first

to reduce cognitive load, even if it results in requiring more time and physical effort [37]. Specifically in our pilot studies, we observed that some users preferred performing the high-effort (mental or physical) actions at the start of the assembly, while others preferred to start with low-effort actions.

If a worker prefers to perform high-effort actions at the end, they must be receiving a higher internal reward for performing the high-effort action at the end instead of at the start. We consider this as the *perceived effort* ε_b of an action based on the user's preference to backload the high-effort actions. To model the time-dependency, we introduce a variable - phase $\psi : s \mapsto [0, 1]$ which represents the percentage of the task that has been completed. We use phase instead of the actual time steps for generality, since the actual task is typically much longer than the canonical task. Using this feature, we model the perceived effort as a linear function of the phase: $\varepsilon_b(a, \psi) = \psi\ \varepsilon(a)$.

We can see that at the start, i.e., $\psi \simeq 0$, the perceived effort will be very small compared to at the end ($\psi \simeq 1$). Thus to maximize the accumulated reward, users will backload the high-effort actions. We can also have the opposite scenario, where a workers prefers to perform the high effort actions at the start. In this case, the reward that the user receives for performing the high-effort actions at the start must be higher than at the end. Hence, we model the perceived effort for frontloading high-effort action as: $\varepsilon_f(a, \psi) = (1 - \psi)\ \varepsilon(a)$.

As our state contains the information of the latest action, we can model the features for frontloading and backloading actions with high physical effort as:

$$\phi_{f,p}(s) = \psi_f(s)\varepsilon_p(s) \tag{3.3}$$

$$\phi_{b,p}(s) = \psi_b(s)\varepsilon_p(s) \tag{3.4}$$

Where, $\psi_f = 1 - \psi$ and $\psi_b = \psi$. Similarly, we can calculate the features for sequencing actions based on their mental effort to obtain the following list of features:

Table 3.2: Effort-based features.

Feature	Weight	Equation	Preference
$\phi_{f,p}$	$w_{f,p}$	$\psi_f \varepsilon_p$	Frontloading of high ε_p actions
$\phi_{f,m}$	$w_{f,m}$	$\psi_f \varepsilon_m$	Frontloading of high ε_m actions
$\phi_{b,p}$	$w_{b,p}$	$\psi_b \varepsilon_p$	Backloading of high ε_p actions
$\phi_{b,m}$	$w_{b,m}$	$\psi_b \varepsilon_m$	Backloading of high ε_m actions

We assume that the effort values for actions in both the canonical and actual assembly tasks can be measured prior to task execution, e.g., through user surveys. Therefore, we use the six features from Tables 3.1 and 3.2 to create our feature function $\phi(s)$, that maps each state s in the canonical and actual tasks to a 6-dimensional feature space. Our goal is to learn the weights for the task-agnostic features from user demonstrations in the canonical task. For example, if a user prefers to backload the high mental effort actions, the weights $w_{b,m} > w_{f,m}$.

3.5 User Study

We want to show that human preferences learned from demonstrations in a canonical task can be used to anticipate their actions in an actual assembly task. Thus, we conduct a user study where participants demonstrate their preferred sequence of actions in a canonical task and an actual model-airplane assembly. We use the demonstrations in the actual task as ground truth to measure the accuracy of anticipating actions based on the weights (i.e., preference) learned from demonstrations in the canonical task.

3.5.1 Actual assembly task

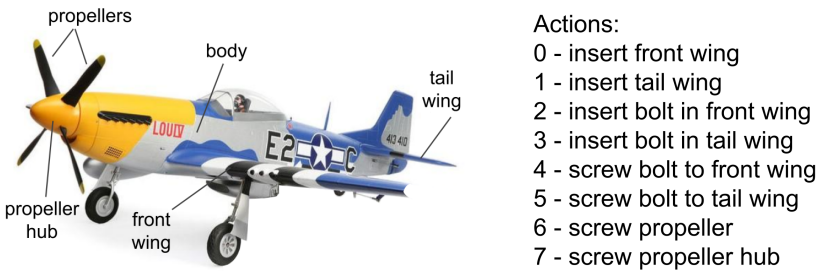


Figure 3.3: Actual assembly task: (Left) Airplane model and (Right) task actions.

We choose an RC model-airplane assembly (see Fig. 3.3) as our actual task. The actions in this assembly task can be sequenced in multiple ways, with few constraints: actions 0, 1, and 6 must precede actions 2, 3, and 7, respectively, and actions 4 and 5 must succeed actions 2 and 3, respectively. The actions can also require different physical and mental efforts. For example, the user shown in Fig. 3.7 rates the action 0 of inserting the (large) main wing as having higher physical effort than the action 6 of screwing a (small) propeller. As some actions need to be repeated (i.e., actions 2, 4, and 6 must be performed 4 times each for inserting 4 bolts in the front wing and screwing 4 propellers), the length of a demonstration in the actual task is 17 time steps. On average, users required 8.81 minutes to complete this task.

3.5.2 Canonical assembly task

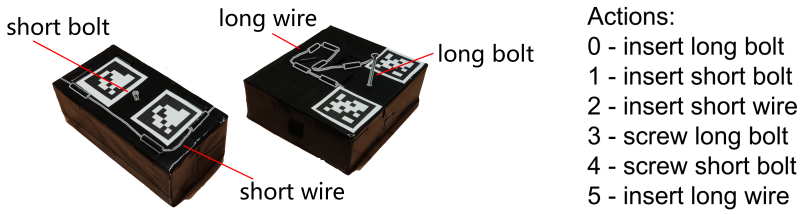


Figure 3.4: Canonical assembly task: (Left) Model and (Right) task actions.

A key challenge in designing the canonical assembly task is to create a set of diverse states and actions, such that we can capture the user preference over a wide range of feature values, while keeping the task significantly shorter than the actual assembly task.

To capture user preferences for consecutively performing actions related to the same part, we design our canonical assembly with two different parts (see Fig. 3.4), each of which is required by at least two different actions. Next, to capture user preferences for consecutively performing actions that need the same tool, we design a pair of actions that require the same tool, i.e., a screwdriver, but have to be performed on different parts. Thus, the users would have to choose between keeping the same tool or keeping the same part when performing the canonical task.

Finally, to capture user preferences for sequencing actions based on their physical and mental effort, we design an action for each combination of high and low values of physical and mental effort:

Table 3.3: Actions with distinct physical and mental efforts.

	Low ε_p	High ε_p
Low ε_m	Action 4	Action 3
High ε_m	Action 2	Action 5

To induce the corresponding physical and mental efforts, we design the actions as follows:

- Action 4, where the user screws a short bolt with a screwdriver. Because the bolt is short, it requires less physical effort to screw it in. Also, since we expect our participants to be familiar with using a screwdriver, we expect this action to require less mental effort.
- Action 3, where the user screws a long bolt using a screwdriver. Because the bolt is long, it requires high physical effort to completely screw it in.
- Action 2, where the user inserts a wire through three small spacers. Because it requires more focus to maneuver a thin wire through narrow passages, we expect it to require high mental effort.
- Action 5, where the user inserts a long wire through six spacers. Since there are more spacers, it requires more physical and mental effort to maneuver the long wire.

We conducted pilot studies to fine-tune the design of the canonical task and verified that participants perceived the physical and mental efforts of the actions as intended. Our final canonical task has 6 time steps, and on average, users required only 3.83 minutes to complete the task. In Chapter 5, we formalize the process of designing such a canonical task for any given actual task.

3.5.3 Study protocol

We recruited 19 ($M = 12$, $F = 7$) participants from the graduate student population at the University of Southern California and compensated them with 20 USD each. The study was approved by the Institutional Review Board (IRB) at our university.

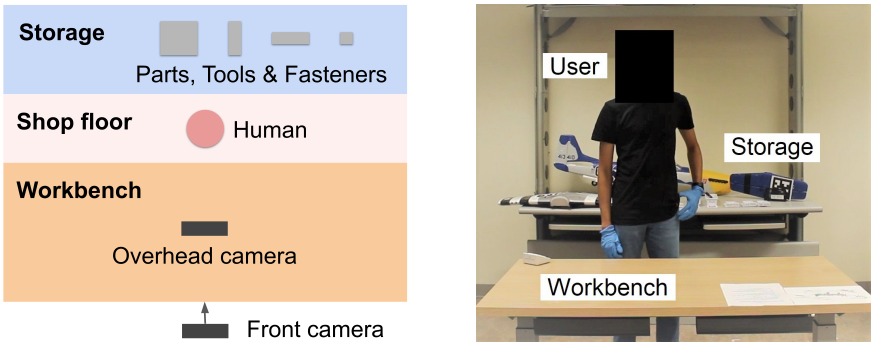


Figure 3.5: (Left) Experimental setup, and (Right) setup at start of actual task

3.5.3.1 Experimental setup

We divide the space into: (i) a storage area where the parts, tools and fasteners are initially placed, (ii) a workbench upon which the user performs all the actions required to complete the assembly, and (iii) the shop floor where the user can stand and move (see Fig. 3.5). April Tags [82] are used to track the parts during the assembly with the help of an overhead camera.

3.5.3.2 Study procedure

We asked each user to perform both the canonical and the actual assembly tasks. We counterbalanced the order of the tasks to guard against any sequencing effect. For each task, we provide a (i) training round - where users learn the assembly task, and an (ii) execution round - where users plan, demonstrate, and explain their preference.

In the training round, we provide each user with a labeled image of the assembled model, as shown in Fig. 3.3 and 3.4, and we describe all the parts and actions in the assembly. We show how to perform

each action in a random order and provide no instructions about the sequence of actions. We also allow users to try each action once for practice, after which they fill in a *post-training questionnaire* to rate (on a scale of 1 to 7) the physical and mental effort that they required for performing each action. We obtain the user's values for ε_p and ε_m for each action from these ratings.

In the execution round, we first give users 5 minutes to plan their preferred sequence of actions for assembling the model as fast as possible. Finally, they demonstrate their preferred sequence and then fill in a *post-execution questionnaire* (see Table 3.4) to report and explain the features that informed their preferred sequence.

Table 3.4: Post-execution questionnaire.

Q1. Explain why you performed actions in the sequence you demonstrated.
Q2. Explain if and how each of the following factors affected your preference:
(i) Physical effort required to perform actions
(ii) Mental effort required to perform actions
(iii) Familiarity with actions
(iv) Perceived importance of an action to the assembly
Q3. Explain and list any other factors that you took into account.

Note: To avoid influencing the users' preference, we do not inform them that their preference in one task will be used to infer their preference in another and we label the tasks as A and B (not canonical and actual). We also do not tell the users to consider effort or movement while planning their preferred sequence. We simply ask them to demonstrate their preferred sequence of actions in each assembly task.

3.6 Experimental Evaluation

We wish to show that user preferences transferred from the canonical task can be used for action anticipation in the actual assembly task. Our hypothesis is that the accuracy of predicting the next user action in the actual task based on their weights learned in the canonical task would be higher than: (i) randomly picking the next action (**H1**) and (ii) randomly setting the weights for the features (**H2**).

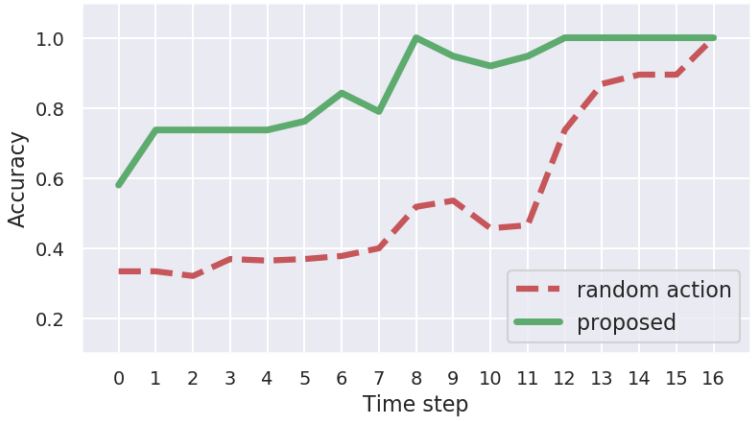


Figure 3.6: Mean accuracy of predicting the user actions at each time step (averaged over all users) in the actual assembly task using our proposed approach (in green) compared to the baseline (in red).

We calculate the accuracy of anticipating the users' actions by comparing the action a_t taken by each user in the actual task with the action $\hat{a}_t$ predicted by our approach at each time step t. The accuracy is 1 when $a_t = \hat{a}_t$, and 0 otherwise.

For baseline (i), we randomly select an action from the remaining actions that can be executed at each time step. For each user, we run the baseline for 100 trials and compute the average accuracy at each time step. We then compare the mean accuracy over all time steps of the proposed method to randomly selecting actions. A two-tailed paired t-test showed a statistically significant difference ($t(18) = 13.82$, $p < 0.001$) between the mean accuracy for our proposed approach ($M = 0.866$, $SE = 0.032$) and the mean accuracy for random actions ($M = 0.543$, $SE = 0.057$). This supports **H1**.

In baseline (ii), we calculate $\tilde{R}_X$ by uniformly sampling random weights for the actual task features and predict actions based on the computed rewards as in the proposed method. Similar to (i), we run 100 trials and compute the average accuracy. A two-tailed paired t-test showed a statistically significant difference ($t(18) = 2.93$, $p = 0.008$) between the mean accuracy for our proposed approach and the mean accuracy for random weights ($M = 0.828$, $SE = 0.042$). This supports **H2**.

Fig. 3.6 shows the mean accuracy of predicting the action at each time step. We can see that the prediction accuracy increases as we reach the final time step, as there are fewer actions to choose at the

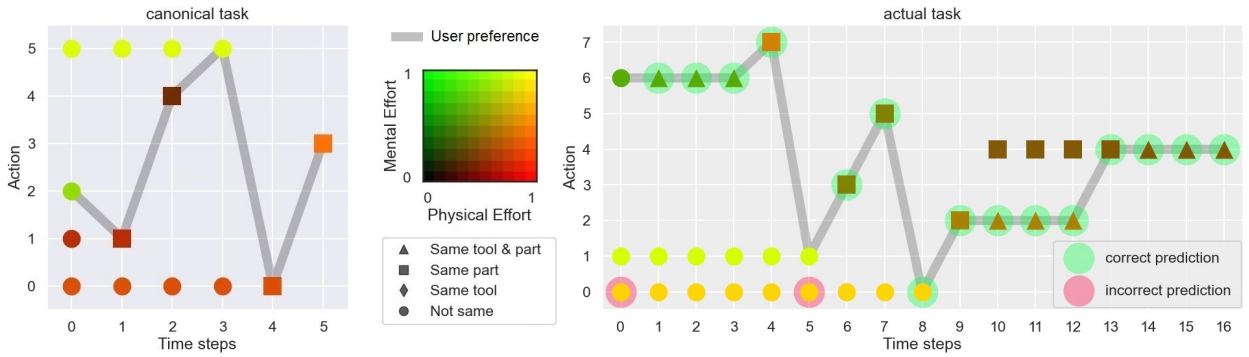


Figure 3.7: Example of a user that prefers to keep working on the same part (□), and perform actions with high physical effort (red) at the end of the task. In the above plots, the x-axis represents the steps (progress) in the task, and y-axis represents all the different actions in the task. At each time step, the actions that can be executed are marked with a shape that indicates whether that action requires the same tool and part as the previous action (refer to legend). The color of the shape indicates the physical and mental effort required to perform that action (refer to color map). In the canonical task, the user performs action 2 at the start of the task (time step 0) because it requires less physical effort (least red) compared to the other choices (actions 0, 1, and 5). At the next time step, the user performs action 1 because it requires the same part as the previous action 2. In the remaining steps, the user performs actions on the same part as before, and if no actions use the same part, perform the least physical effort action. Interestingly, the user follows the same preference in the actual task by performing the least physical effort (least red) action 6 at the start. The predictions made by our proposed approach at each time step are shown with a larger green (or red) circle. We see that our proposed approach is able to accurately predict the user's actions at time steps 1 and 4 when the user keeps working on the same part.

end. The accuracy for random actions (and random weights) is 0.33 at the time step 0 as there are 3 actions that can be performed at the start. The accuracy for our proposed method is significantly higher at the start as we correctly anticipate the first action for 11 out of 19 users based on their transferred weights. After the first action, the accuracy for our proposed method improves further as we are able to predict subsequent actions based on user preferences for minimizing part and tool change.

3.7 Discussion

We consider two user examples to demonstrate how preferences transfer from our canonical to actual task.

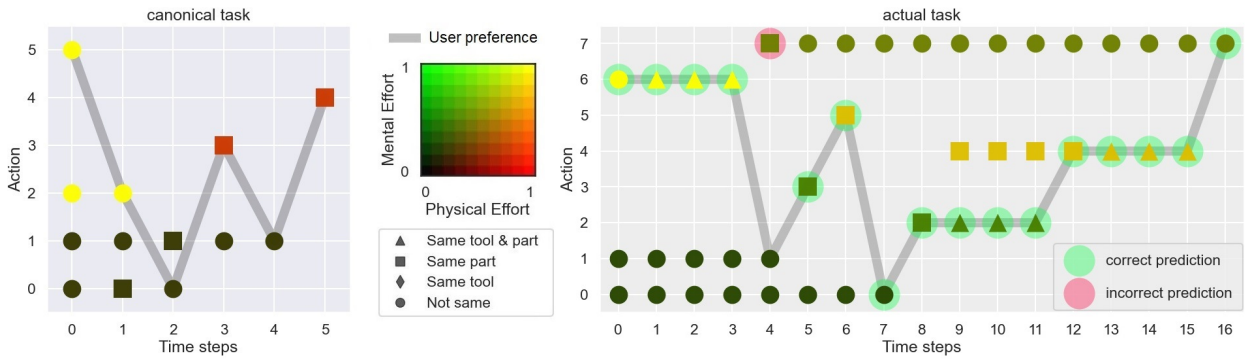


Figure 3.8: Example of a user that prefers to perform actions with the high mental effort at the start of the task. In the canonical task, the user performs actions 5 and 2 that have high mental and physical effort (yellow) at the start (time steps 1 and 2). While the user leaves actions 3 and 4 that have low mental effort and high physical effort (red) for the end (time steps 3 and 5). Thus, from the canonical task demonstration, we learn that the user prefers to frontload actions with high ε_m. In the actual task, the user starts with action 6 that has the highest mental effort. Based on the canonical task, our proposed method correctly predicts (green circle) that the user will perform action 6 at time step 0. Similarly, at time step 5 the user performs action 3, which is also what we predict, since it has higher mental effort than actions 0 and 7.

3.7.1 Learning user preferences in the canonical task

Consider the demonstrations shown in Fig. 3.7, where the user prefers to pick the actions with the same part as the previous action whenever possible (time steps 1, 2 and 4). In the other steps, the user picks the action with the least physical effort (time steps 0 and 3). Accordingly, the weights we learn from the canonical task demonstration are higher for the feature of keeping the same part ($\phi_{\mathcal{P}}$), followed by the feature of backloading actions with high physical effort ($\phi_{b,p}$).

On the other hand, consider the user in Fig. 3.8 who prefers to perform actions with high mental effort (actions 5 and 2) at the start of the task (time steps 1 and 2). Based on their demonstration, we learn a high weight for the feature of frontloading actions with high mental effort ($\phi_{f,m}$).

We note that none of the users had the exact same weights w_C even if some users had the same demonstration sequence ξ_C, since they gave slightly different ratings for the physical and mental effort of the canonical task actions.

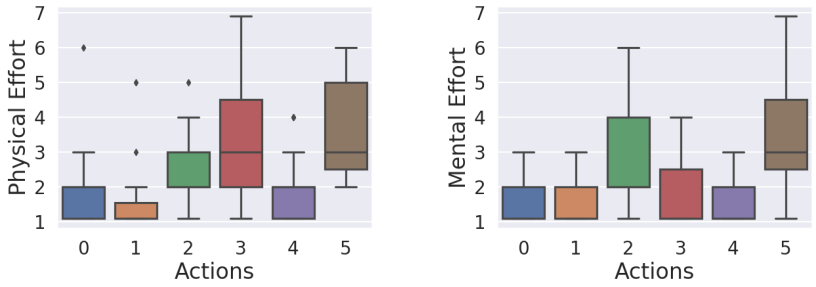


Figure 3.9: Mean user ratings for the physical and mental effort of actions in the canonical task.

3.7.2 Transferring learned preferences to the actual task

For many users, weights learned for the task-agnostic features in the canonical task enable accurate anticipation of their actions in the actual task. For the user in Fig. 3.7, we learn high weights for keeping the same part ($\phi_{\mathcal{P}}$) and backloading actions with high physical effort ($\phi_{b,p}$). Using these weights to calculate rewards in the actual task, we can accurately anticipate the actions at 15 out of the 17 time steps. For example, we correctly anticipate action 7 at time step 4 as it uses the same part as the action 6 at the previous time step. Similarly, for the user in Fig. 3.8, we are able to transfer their preference for frontloading actions with high mental effort. For example, at time step 0, we accurately predict the user's action as 6, since it has the highest mental effort.

However, in a few cases, preferences over a specific feature did not transfer to the actual task due to human variability. For example, we incorrectly predict at time steps 0 and 5 for the user in Fig. 3.7 since we also learn a high weight for backloading actions with high mental effort ($\phi_{b,m}$) in the canonical task. This is because the last three actions in the user's demonstration have a higher mental effort than the first three actions. Therefore, in the actual task, we predict action 0 at time step 0 even if it has high physical effort (low immediate reward), since it would allow the user to perform subsequent low mental effort actions (not shown in the figure) at the start of the task. However, since the user's true preference is to only backload the high physical effort actions, the user performs action 6 instead. Thus, we see that while the user retains

their preference for keeping the same part and backloading actions with high physical effort in the actual task, their preference for backloading actions with high mental effort in the canonical task is not retained.

Finally, we discuss a limitation where the user preference in the actual task was affected by a new feature that wasn't modeled in the canonical task. For example, in Fig. 3.8, we expect the user to perform action 7 at time step 4 based on their preference for frontloading actions with high mental effort. However, we see that the user instead performs action 1 which has lower mental effort and leaves action 7 for the end. Based on open-ended responses provided by users at the end of the study, we suspect that the user preferred to leave the action of screwing the propeller hub (action 7) for the end as they thought that the propellers might break (when they perform the remaining actions) if they were attached at the start.

While most users stated that they predominantly considered the same features as described in Section 3.4, some users commented that they also considered *"not want[ing] parts to break"* or that the *"airplane body will [also] take up space so do propellers first"*. Thus, their preferred sequence in the actual task was affected by new features for 'part breakage' and 'space required' that we had not considered. In such cases, we would like the robotic assistant to identify when a new feature affects user actions and query the user to actively learn the weights for the new feature. We address this limitation in the next chapter.

Overall, in this study, we see that preferences can be transferred when the features that determine the user's preferred sequence in the canonical task overlap with the features considered by the user to determine their preference in the actual task. We provide a video at youtu.be/o-_GJWG9NhQ that explains our approach for the user depicted in Fig. 3.8.

3.8 Conclusion

Our work demonstrates the potential of anticipating user actions in actual assembly tasks based on preferences learned over task-agnostic features in shorter canonical tasks. Our results show that predictions in the actual assembly task based on transferred preferences are significantly better than the predictions

for random action selection and random weights. Moreover, by obtaining user demonstrations in a shorter canonical task instead of the actual assembly, we can reduce the time and human effort required to learn user preferences.

In the future, we want to evaluate the benefit of transferring human preferences from canonical to actual assembly tasks by having a robotic assistant perform the anticipated actions. We would also like to account for users that change their preferences when switching from the canonical to the actual task. In the next chapter, we extend our approach to update the transferred preferences online during the actual task and evaluate the benefit of proactively assisting users in a human-robot assembly study.

While the current setup assumes full observability of the workspace by both the robot and the user, future work should consider sensor placement [81] and user viewpoint [77] in the robot's decision making. Finally, automatically inferring the effort for different actions, instead of using questionnaire responses, is an important area for future work.

Chapter 4

Updating Transferred Preferences Online for Proactive Assistance

4.1 Introduction

We want robots to proactively assist users in assembly tasks by predicting their actions [49, 57, 51]. 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 improve human productivity by reducing the time for which the user remains idle [45].

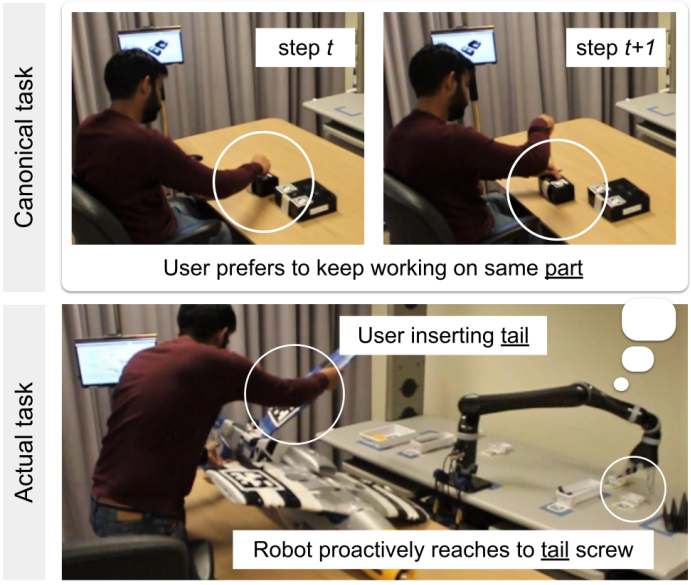


Figure 4.1: We use task-agnostic preferences in the canonical task, such as consecutively perform all actions that use the same part, as a prior for predicting user actions in the actual task. A robot uses the predictions to proactively assist users, while also updating the prior through interaction.

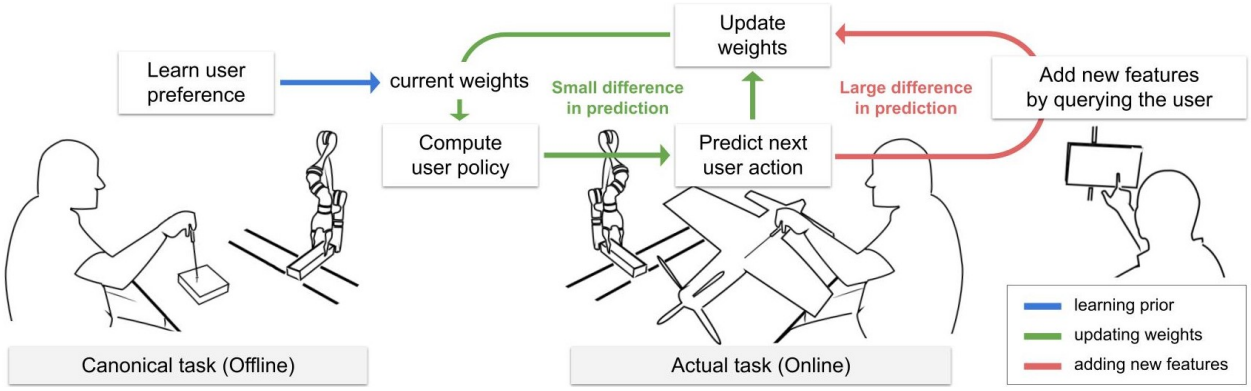


Figure 4.2: Proposed system for transfer learning of human preferences: In the offline phase, the robot learns the preference of a given user from their demonstrations in a canonical task. The preference is encoded as weights of task-agnostic features that constitute the user's reward function and is 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 their reward function (red).

As in the previous chapter, we focus on the problem of predicting user actions in an actual assembly by efficiently learning the user preferences, without any demonstrations in the actual task. One class of approaches, as proposed in Chapter 2, learns dominant preference models by clustering demonstrations of a population of users in the actual task [79, 75] and then associates the preferences of new users with the learned dominant models. While this removes the need for demonstrations from the new user, it requires several prior demonstrations on the task. Alternatively, users can observe robot actions and provide online corrections during task execution [70, 16]; however, an inaccurate initial preference model may result in a large number of user interactions for correcting the robot.

How can we obtain an accurate initial preference model without user demonstrations in the actual task?

In the preliminary study in Chapter 3, participants explained their action selection in a model-airplane assembly 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 (see Fig. 4.1). 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 [74]. In our example, the canonical task includes actions with varying physical and mental efforts, and allows users to choose between keeping the same part and tool or switching them, so that we can learn a preference model with respect to these task-agnostic 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 [37]. Furthermore, users may consider new features when deciding their preference in the actual task that were not modeled when learning their preference in the canonical task, e.g., the space occupied by the parts in the actual task.

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. 4.2). 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 the user preference cannot be accurately predicted with the existing set of features.

4.2 Related Work

4.2.1 Learning user preferences offline

While user preferences can be modeled as constraints in a task scheduling problem [40], defining such constraints can be challenging for end users. Hence, preferences are often learned implicitly from demonstrations of the user's preferred behaviour [84, 120]. One such approach predicts user actions based on the frequency of observing the action transitions in the user's demonstrations [84]. This limits the accuracy of prediction to only those transitions observed in the demonstration data. Instead, inverse reinforcement learning (IRL) [1] learns a reward function that can reproduce the demonstrated user sequences [110] and generalize to other states in the task.

To efficiently infer user preferences, demonstrations of several users can be clustered to learn dominant preference models, such that the preference of a new user can be quickly inferred by matching their actions to a dominant model [79, 75]. However, this requires demonstrations from a large population of users to cover the range of preferences that we expect the users to have in the actual task.

Since user demonstrations can be difficult to obtain in real-world assembly tasks, an alternative is to obtain the user's preferred choice from a set of uniformly sampled [105] or actively generated trajectories [94]. To efficiently converge to the user preference, the robot can generate the trajectories such that any selection made by the user will remove the maximum volume (i.e., uncertainty) from the hypothesis space. In addition to minimizing the robot's uncertainty over the user's preference, we can also minimize the user's uncertainty over the trajectories to generate queries that are easier for the user to answer [17].

But this work only focuses on learning user preferences for the robot's trajectory instead of learning to predict user actions, as in our scenario.

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 [16]. However, this would still require the demonstrations to be obtained in the actual task prior to task execution.

To reduce the human effort in providing demonstrations, previous work has also explored transferring human strategies in simulated search-and-rescue tasks from simple to more complex environments [44]. Similarly, in an IKEA assembly study without a robotic assistant, our preliminary work [74] has shown that user preferences learned in a canonical task are useful for predicting human actions in an actual assembly. The work assumes that user preferences do not change when switching from the canonical to the actual task and that user preferences in both tasks depend on the same set of features. However, we show that the robot cannot solely rely on a transferred estimate of user preference in the actual task, e.g., because of a change in their preference or because of features that were missing 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.

4.2.2 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 [70]. 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 [42]. Similarly, human interventions can also be used to directly update the parameters of the quality function via behaviour cloning [65].

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 [55, 8, 60]. The robot is initialized with a reward function that is a weighted combination of task features and the weights are updated based on the difference in feature counts of the 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 [98]. Similarly, user feedback for right or wrong actions can be used as a reward signal for Q-learning [61, 53, 63].

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, our proposed system initializes the preference model with a prior learned from a canonical task to efficiently adapt to the user during the actual task.

4.3 System for Transfer Learning of Human Preferences

As in Chapter 3, we model each task as a Markov Decision Process (MDP) defined by $M := (S, A, T, R)$; where S is the finite set of states in the assembly, A is the finite set of assembly actions that must be performed by the user to finish the assembly, $T(s_{t+1}|s_t, a_t)$ is the probability of transitioning from state s_t to s_{t+1} by taking action a_t, and $R(s_{t+1})$ is the reward received by the user in s_{t+1}. We assume that S, A, and 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 perform value iteration [12] to compute the user's policy $\pi(s_t)$ that maps each state $s_t \in S$ to a human action $a_t \in A$. 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 4.5).

Algorithm 3 Updating transferred preferences online

Require: Actual task M_X, transferred weights w_C, task-agnostic features ϕ, candidate features 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
    $\hat{a}_t = \pi_t(s_t)$
    Observe a_t
    if $a_t \neq \hat{a}_t$ **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 $\hat{\Xi}_{1:T}$
        $w_{X,t+1} = maximumEntropyIRL(\hat{\Xi}_{1:T}, w_{prior}, \phi_{t+1})$
        $R_{X,t+1} = w_{X,t+1} \cdot \phi_{t+1}$
        $\pi_{t+1} = valueIteration(R_{X,t+1}, M_X)$

Our system consists of two phases: (i) an offline phase where the robot learns the user preference in a 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 3).

4.3.1 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:

$$R_C(s) = w_C^T \phi(s) \;\; \forall s \in S_C, \quad R_X(s) = w_X^T \phi(s) \;\; \forall s \in S_X \tag{4.1}$$

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 [121, 110] 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.

$$\nabla \mathcal{L}(w_C) = \frac{1}{|\Xi_C|} \sum_{\xi_C \in \Xi_C} \sum_{s \in \xi_C} \phi(s) - \sum_{s \in S_C} D_\pi(s)\phi(s) \tag{4.2}$$

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:

Motivating example. Consider two canonical task states $s_{1,C} = \{\text{long wire inserted}\}$ and $s_{2,C} = \{\text{screw inserted}\}$ and a task-agnostic feature function: $\phi(s_C) = [\phi_1(s_C), \phi_2(s_C)]$ where ϕ_1 is the feature for mental effort and ϕ_2 is for physical effort. Let $\phi(s_{1,C}) = [0.9, 0.1]$ and $\phi(s_{2,C}) = [0.1, 0.8]$, because $s_{1,C}$ corresponds to high mental effort, while $s_{2,C}$ to high physical effort. The human preference in the canonical task is encoded with a reward function $R_C(s_C) = w_C \cdot \phi(s_C)$, where $w_C = [w_{1,C}, w_{2,C}]$. We learn w_C from user demonstrations. If the user prioritizes mental effort over physical effort, $w_{1,C} > w_{2,C}$. Let us assume $w_C = [1, 0]$.

Let two actual task states $s_{1,X} = \{\text{propeller assembled}\}$, $s_{2,X} = \{\text{wing inserted}\}$, and $\phi(s_{1,X}) = [1, 0]$ and $\phi(s_{2,X}) = [0, 1]$, so that $s_{1,X}$ is high mental effort and $s_{2,X}$ high physical effort. While states in the canonical task (S_C) and actual task (S_X) are different, the features ϕ are common across both tasks. We encode the human preference in the actual task with $R_X = w_X \cdot \phi(s_X)$. Our system uses w_C to initialize w_X, so that $w_X = [1, 0]$. We then have $R_X(s_{1,X}) = 1$ and $R_X(s_{2,X}) = 0$. A policy from some starting

state that maximizes R_X would map to an action that leads to the high mental effort state $s_{1,X}$, e.g., 'assemble propeller'. We use the policy to predict human actions in the actual task.

4.3.2 Updating feature weights

To account for the changing user preferences from canonical to actual tasks, 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 $\hat{a}_t$ 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 Algorithm 3). If the action performed by the user a_t does not match our prediction $\hat{a}_t$, i.e., $a_t \neq \hat{a}_t$, we update the weights as follows:

By the time step t of the actual task, we have only observed the user's action sequence $\xi_{1:t}$ up to that point. Computing the weights based solely on $\xi_{1:t}$ may be insufficient for inferring the user's preference for the rest of the task. So, 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. Thus, $\xi_{1:t}$ records the user's current preferences in the actual task while $\hat{\xi}_{t+1:T}$ is based on the robot's previous estimate of the user preference.

Similar to the offline setting, we use maximum-entropy IRL [121] 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 [8, 92].

4.3.3 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 [21]. 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 [39].

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 pre-defined threshold Δp_{max} (step 10 in Algorithm 3).

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 4.3.2. We also remove that feature from the set of candidate features ϕ_X. If the user does not select any of the candidate features, we update the weights using the existing feature set. Fig. 4.13 shows an example of the query presented to a participant in our user study.

Table 4.1: Simulated user weights.

weights	w_1	w_2	w_3
1	0.6	0.2	0.2
2	0.8	0.1	0.1
3	0.2	0.6	0.2
4	0.1	0.8	0.1
5	0.2	0.2	0.6
6	0.1	0.1	0.8
7	0.4	0.4	0.2
8	0.4	0.2	0.4
9	0.2	0.4	0.4
10	0.4	0.3	0.3
11	0.3	0.4	0.3
12	0.3	0.3	0.4
13	0.6	0.3	0.1
14	0.6	0.1	0.3
15	0.3	0.6	0.1
16	0.1	0.6	0.1
17	0.3	0.1	0.6
18	0.1	0.3	0.6
19	0.25	0.35	0.40
20	0.33	0.33	0.33

4.4 Simulation Experiments

We first evaluate our proposed system with simulated users 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 accuracy of predicting user actions.

We consider a canonical task (M_C) with $|A_C| = 6$ actions and $|S_C| = 27$ states and an actual task (M_X) with $|A_X| = 10$ actions, and $|S_X| = 243$ states. We model T_X and T_C such that each action can only be executed once with some actions having ordering constraints. Specifically, for a task with N actions, we specify that each action a_i can only be formed if action a_j has been performed before, where $i = [N/2 + 1, \ldots, N]$ and $j = i - N/2$.

Table 4.2: 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

We assume a 3-dimensional feature space Φ to model the reward function. Details of a similar feature function can be found in the following chapter in Section 5.5.1.2. We manually design 20 simulated users with substantially different preferences (see Table 4.1), since we found that sampling the weights uniformly at random resulted in a small number of distinct preferences.

We individually evaluate the three components of our proposed system by referring to our approach of transferring user preferences as $prior$, updating the transferred weights as $online$ (Section 4.3.2) and adding new features as $online\_add$ (Section 4.3.3). Table 4.2 shows the differences in each approach.

4.4.1 Benefit of transferring weights from the canonical task

We first compare predicting user actions with online update of the transferred weights (*online*) and randomly initialized weights (*rand_online*) to show that weights transferred from a canonical task provide a better initial estimate of user preferences in the actual task.

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 simulate the user policy in both tasks. (2) `opposite`: Users have a very different preference in the actual task, compared to the canonical task. We simulate the user 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 $\hat{a}_t$ to the simulated actions a_t in the actual task. The accuracy is 1 when $a_t = \hat{a}_t$, and 0 otherwise. For each user, we compute

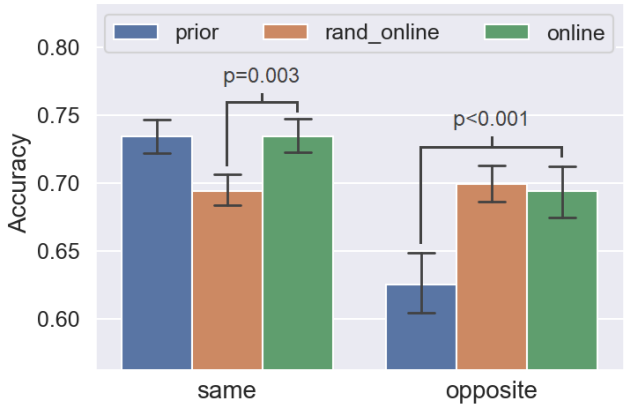


Figure 4.3: Mean accuracy of predicting the simulated user actions in the actual task for the `same` and `opposite` scenarios. The error bars indicate the standard error.

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. 4.3 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 mean accuracy of $online$ ($M = 0.73$, $SE = 0.012$) and $rand\_online$ ($M = 0.69$, $SE = 0.012$). On the other hand, we do not see a significant difference for users in the `opposite` scenario.

4.4.2 Benefit of updating feature weights in the actual task

We compare using the transferred weights with (*online*) and without online updates (*prior*) to show that it is important to adapt to the changed user preferences for making accurate predictions. In the `opposite` scenario, where users change their preference, a two-tailed paired t-test shows a statistically significant difference ($t(19) = 5.17$, $p < 0.001$) between the mean accuracy of *online* ($M = 0.69$, $SE = 0.017$) and *prior* ($M = 0.62$, $SE = 0.023$). When comparing to the `same` scenario, we see that the mean accuracy of $online$ reduces by only 5.47% in `opposite`, while the mean accuracy of $prior$ reduces by 14.87%.

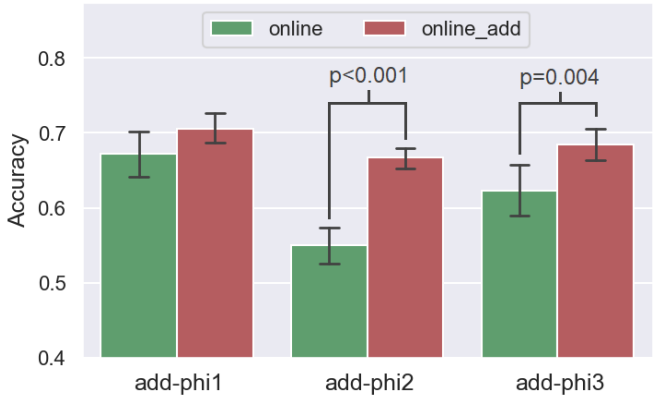


Figure 4.4: Mean accuracy of predicting the simulated user actions using our proposed approach with and without feature addition for $\Delta p_{max} = 3$.

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.

4.4.3 Benefit of adding new features in the actual task

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 learning their preference in the canonical task, we exclude the first feature ϕ_1 and only learn the weights for the remaining two features. The excluded feature is added to the set of candidate features and can be integrated into the preference model during the actual task (`add-phi1`). We similarly consider two more cases excluding ϕ_2 (`add-phi2`) and ϕ_3 (`add-phi3`).

We compare the proposed system 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 automatically adding the excluded feature when $\Delta p >$

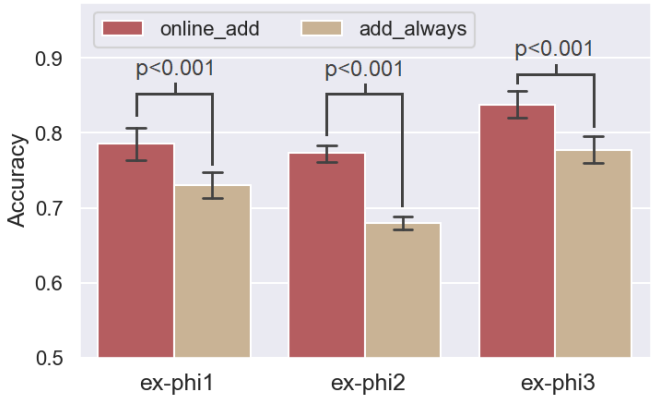


Figure 4.5: Mean accuracy of predicting user actions using our proposed approach with feature addition based on Δp_{max}.

Δp_{max}, without any user selection. In the actual system, the user chooses a relevant feature from the set of candidate features and has the option to not add any feature.

Fig. 4.4 shows that adding new features (*online_add*) leads to a higher accuracy of action prediction in the actual task than just 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* ($M = 0.54$, $SE = 0.02$). Similarly for `add-phi3`, a two-tailed paired t-test shows a statistically significant difference ($t(19) = 3.21, p = 0.004$) in the mean accuracy of *online_add* ($M = 0.68$, $SE = 0.02$) and *online* ($M = 0.62$, $SE = 0.03$).

4.4.4 Benefit of selectively adding new features

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 the 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 similarly consider two more cases excluding ϕ_2 (`ex-phi2`) and ϕ_3 (`ex-phi3`). Fig. 4.5 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.

4.5 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 positively affect the subjective user experience.

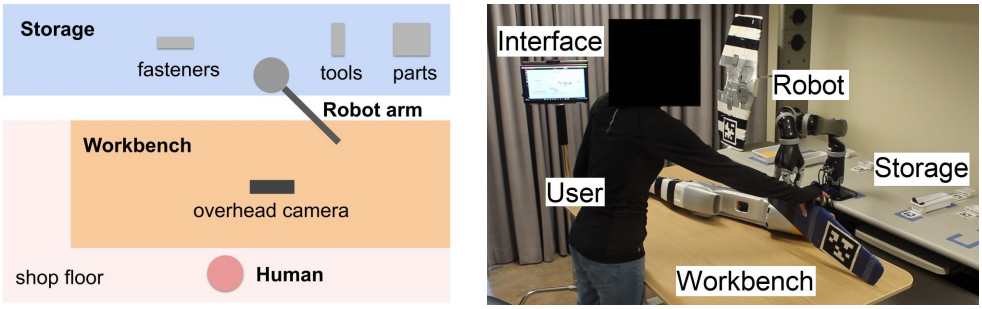


Figure 4.6: (Left) Experimental setup for human-robot assembly, and (Right) setup during the actual task.

4.5.1 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 Fig. 4.6).

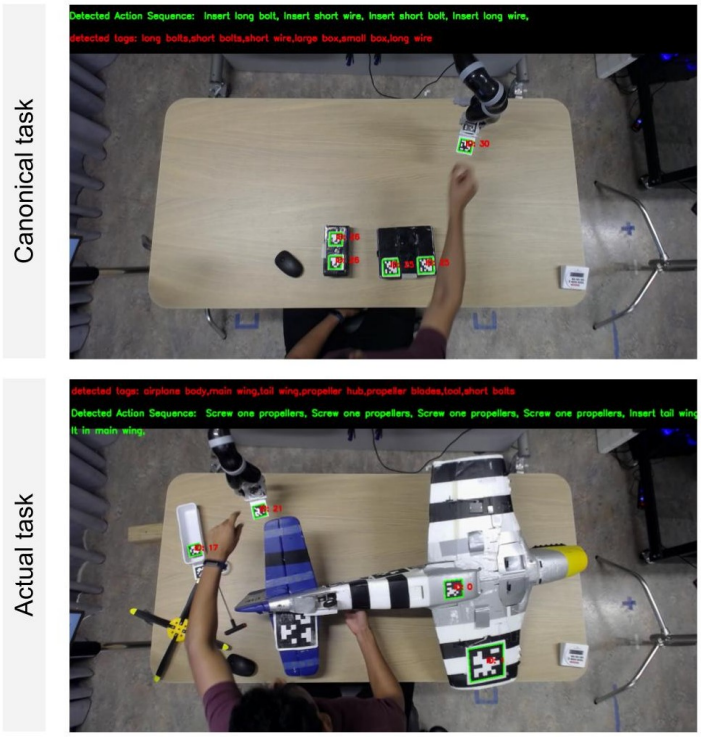


Figure 4.7: Top-down view of the canonical (top) and actual (bottom) tasks showing detected parts.

We program the robot to ask-before-acting [70], where the user confirms the robot's action so that the robot does not incorrectly deliver the wrong part. We use AprilTags [82] to detect the parts and tools present in the workbench (see Fig. 4.7) and we recognize user actions using a manually specified dictionary that maps specific part configurations to assembly operations (see Tables 4.4 and 4.3).

4.5.1.1 Actual and canonical assembly tasks

We use the same actual and canonical tasks as in the previous chapter (see Section 3.5). The actual task is a model-airplane assembly with $|A_X| = 8$ actions, i.e., a_1: insert main wing, a_2: insert tail wing, a_3: insert

Table 4.3: Parts and tools required for actual task actions.

Action	Required parts and tool (hex key)
a_1	main wing, airplane body
a_2	tail wing, airplane body
a_3	long bolt
a_4	tail screw
a_5	tool
a_6	tool
a_7	propeller blade, propeller hub, short bolts, tool
a_8	propeller nut, airplane body

Table 4.4: Parts and tools required for canonical task actions.

Action	Required parts and tool (screwdriver)
a_1	long bolt, large box
a_2	short bolt, small box
a_3	short wire, small box
a_4	tool
a_5	tool
a_6	long wire, large box

long bolt, a_4: insert tail screw, a_5: screw long bolt, a_6: screw tail screw, a_7: screw propeller blade, and a_8: screw propeller hub, and $|S_X| = 3324$ states.

On the other hand, the canonical task consists of only $|A_C| = 6$ actions, i.e., a_1: insert long bolt, a_2: insert short bolt, a_3: insert short wire, a_4: screw long bolt, a_5: screw short bolt, and a_6: insert long wire, and $|S_C| = 175$ states. Users can perform the actions in any order to complete the assembly with the only constraint that parts and bolts must be inserted before screwing. We assume deterministic transitions T_C and T_X in both tasks and set $\Delta p_{max} = 3$ (step 10 in Algorithm 3).

The state in the canonical and actual assembly tasks records the parts that have been assembled and the last two actions performed by the user. We augment the last two user actions to the assembly state to capture user preferences for 'keeping the same part' and 'keeping the same tool'. A terminal state is reached when all the parts have been assembled.

4.5.1.2 Feature space

We specify for the canonical task 6 task-agnostic features from the previous chapter 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 required parts.

4.5.2 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 the next user action (step 7 in Algorithm 3) and proactively reaches the parts required for that action. We assume that the robot has access to the 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. 4.8). 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.

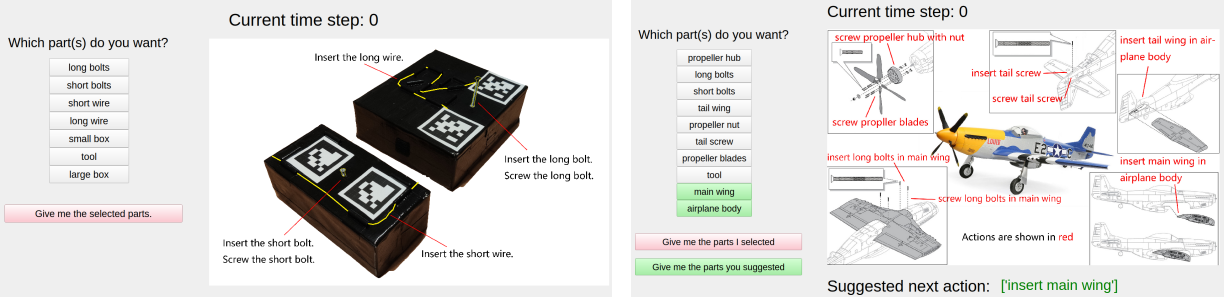


Figure 4.8: Graphical user interface for interacting with the robot in the canonical (left) and actual (right) assembly task. The predicted action and required parts are highlighted in green.

4.5.3 User study protocol

We recruited 18 ($M = 13$, $F = 5$) participants from the graduate student population at the University of Southern California, using a sign-up form sent out through the university mailing lists. Participants were compensated with 20 USD each. 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 the 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 4.5) and answer open-ended questions about their subjective experience with the robot.

4.5.4 Hypotheses

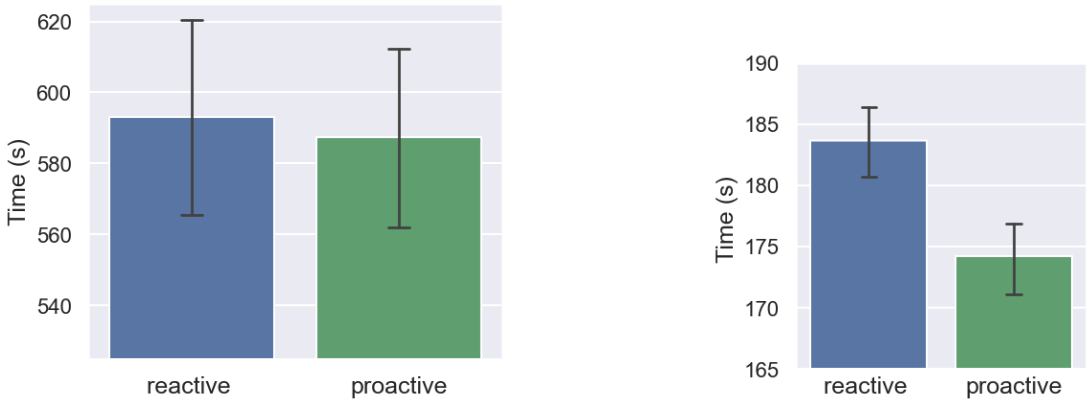
We hypothesize that participants will require less time to complete the assembly when working with the proactive robot P, than with the reactive robot R (**H1**). In addition to task efficiency, we expect that proactively assisting users will improve the human-robot team fluency (**H2**), using human idle time as a team fluency metric. We base this on previous work [49, 79] 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 [48] and we design the remaining questions following recommended practices [97]. 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.

4.5.5 Experimental results

4.5.5.1 Task execution time

We measure the total time required by users to complete the actual assembly task. A two-tailed paired t-test did not show a significant difference in task execution time 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 the participants were not skilled assembly workers. We observed that some users took more time to complete an action as they lost focus



(a) Task execution time

(b) User idle time

Figure 4.9: Average time taken by users to complete assembly and the average time for which the user remains idle in the actual task. The error bars indicate the standard error.

or got tired. On the other hand, some users took less time to perform the actions as they gained more experience on the task.

4.5.5.2 Team fluency

We measure human idle time as the team fluency metric, which is the time spent by the user waiting for the robot to fetch the 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 P condition, when the robot correctly predicted the participant's next action, their idle time decreased because the robot would reach the required parts in advance. On the other hand, if the robot made an inaccurate prediction, it returned to the starting position and the participant 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 4.6), we see the total idle time was reduced. This supports our hypothesis **H2**.

4.5.5.3 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 [33] and report it in the questionnaire in Table 4.5. We treat the combined ratings of all items in a scale as interval data for performing statistical tests [97].

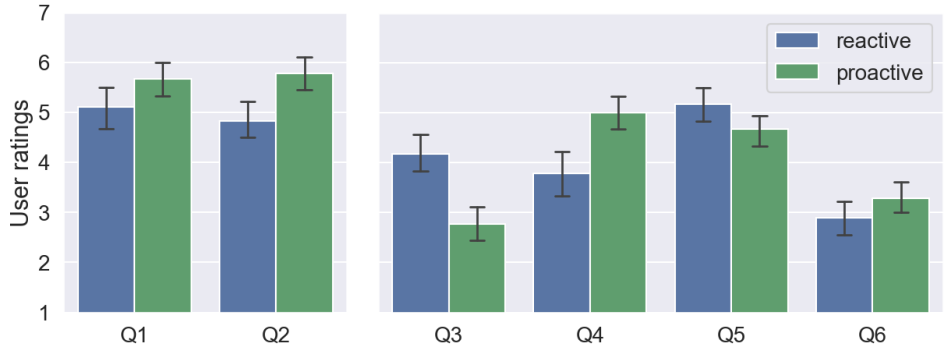


Figure 4.10: User ratings for questions in the perceived fluency ($Q1 - Q2$), and relative contribution ($Q3$ - $Q6$) scales.

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

Table 4.5: 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.

R ($M = 4.97$, $SE = 0.38$), a two-tailed paired t-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 $Q3 - Q6$. A two-tailed paired t-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**. Fig. 4.10 shows the average user ratings for each question in the perceived fluency and relative contribution scales.

Trust. A two-tailed paired t-test did not show a significant difference in the user's trust ($Q7 - Q8$) between P and R conditions, 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 [26].

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 ($M = 3.42$, $SE = 0.42$) conditions. This supports **H6**.

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*", and 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]*".

4.6 Ablation Studies

We use the data recorded in the assembly study of Section 4.5 to explore how each component of the proposed system contributes to prediction accuracy. We define *prior*, *rand_online* and *online* as specified

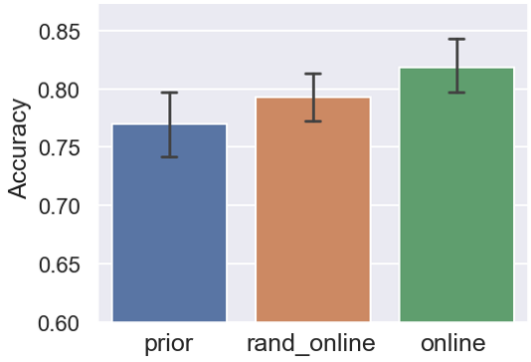


Figure 4.11: Mean accuracy of predicting user actions in the actual assembly by updating their weights online. The error bars indicate the standard error.

in Table 4.2 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 model online are critical for accurately predicting the actions of real users in the actual assembly task.* While the transferred weights offer a better initialization for action prediction, updating the weights online becomes essential for users that change their preference across tasks.

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

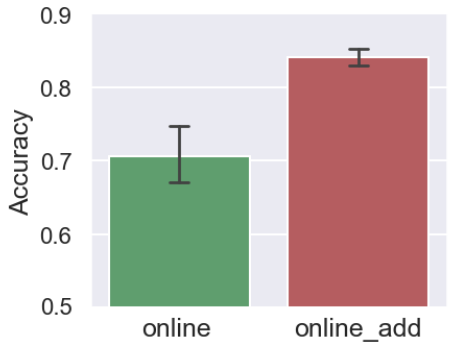


Figure 4.12: Mean accuracy of predicting user actions in the actual assembly using our proposed approach with feature addition ($online\_add$) and without ($online$). The error bars are standard errors.

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, Algorithm 3. 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 4.7.

4.7 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 4.5 as follows: (1) We exclude from the feature set ϕ the feature for 'keeping the same part' and add it to the list of candidate features ϕ_X. (2) In the study of Section 4.5, 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 prioritize the feature for 'keeping the same part'.

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 4.5, but instead of a reactive robot as the

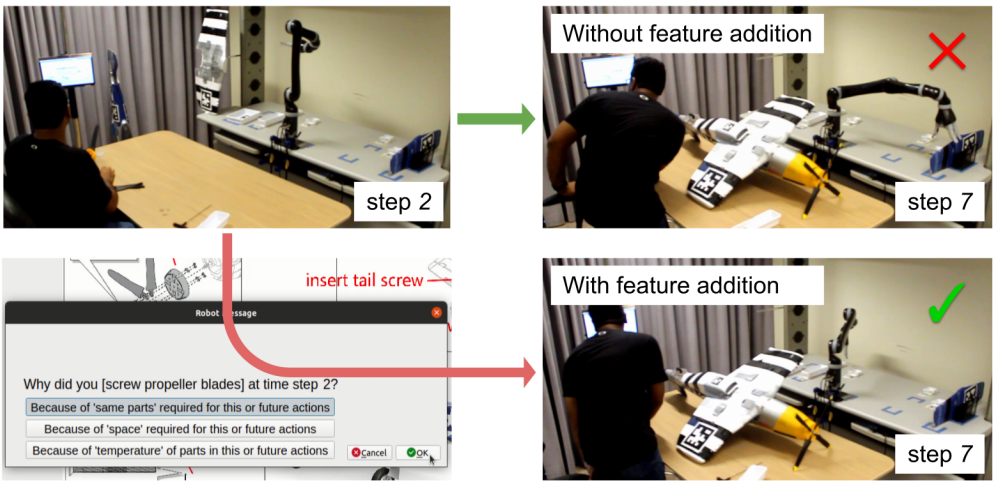


Figure 4.13: The participant selects the feature for 'keeping same part' when the robot incorrectly predicts their action. By incorporating the selected feature, the robot can later correctly predict the user's action when they keep using the main wing of the airplane. However, if the robot does not add the new feature, the updated weights for the existing features cannot accurately capture the user's preference.

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. 4.13 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 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.

We provide a video of how the robot proactively assisted users in the human-robot assembly study and queried users to add new features in the follow-up study at youtu.be/8SO6Zh3wN24.

4.8 Conclusion

While we evaluate the proposed system on a single assembly task, future work should test the transfer of user preferences 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. In Chapter 5, we propose automatically generating the canonical assembly task for a given task-agnostic feature space.

4.8.1 Limitations

In addition to asking users to select relevant candidate features with pre-computed feature values, we could ask users to directly assign [21] or teach [18] 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 [78]. 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 [122].

4.8.2 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 real-world assembly task. While there has been significant work on preference learning from human inputs directly in the actual

task [113], learning from a shorter, expressive task can significantly reduce the burden of time-consuming demonstrations.

Chapter 5

Selecting Canonical Tasks for Transfer Learning of Human Preferences

5.1 Introduction

In the previous chapter, we proposed transferring human preferences from short, canonical assembly tasks to more complex assemblies for predicting user actions [73]. Similarly, related work has shown that human navigation strategies can be learned in a smaller map and transferred to a larger map for efficiently training the robot [44]. Providing demonstrations in a simpler source task can be much easier than in a complex target task, requiring less human effort overall.

However, in these approaches, the source tasks used to learn the prior models of user preference are manually specified by an expert. Moreover, prior work does not investigate how the differences between the source and target tasks affect the efficacy of the preference transfer. If the two tasks are substantially and arbitrarily different, preferences learned in the simpler source task may not be useful for predicting user actions in the actual target task. Thus, in this work, we focus on the following question:

How can we automatically select a good source task, referred to as the canonical task, such that human preferences learned in the canonical task can be transferred to the target task for making accurate predictions?

A key concept in transfer learning of human preferences is to model the user preference over *task-agnostic features* that are shared by the source and target tasks. Specifically, the objective function that dictates user actions in both tasks can be represented as a weighted combination of task-agnostic features,

such as the physical and mental effort required to perform the respective task actions [74]. With this formulation, the robot can learn the feature weights based on user demonstrations in the source task, compute the user's objective function, and use the same objective to predict user actions in the target task.

Therefore, to make accurate predictions in the target task, the robot must (i) identify the relevant task-agnostic features and (ii) be able to learn the actual feature weights for each user from their demonstrations in the source task. In this work, we assume that the robot knows the task-agnostic features that are relevant for capturing user preferences in the target task. Our focus, then, is on identifying an appropriate canonical task that can facilitate accurate learning of feature weights for different users.

The problem of learning user objectives from demonstration data is referred to as inverse reinforcement learning (IRL) [76]. Typically, in IRL, there can be many objectives that produce the same set of demonstrations [5]. In this case, the robot cannot distinguish between users who may have different weights if they provide similar demonstrations because of the task structure. To ensure that the feature weights for different users can be recovered accurately, we would need their demonstrations in the canonical task to be distinct from each other.

Thus, our key insight is that the canonical task should be expressive, i.e., *it should allow different users to uniquely demonstrate their individual preferences.* The main contribution of our work is a metric for measuring the *expressiveness* of source tasks when learning the feature weights via maximum entropy IRL [121]. Our proposed approach obtains a collection of source tasks from a random task generator, scores their expressiveness by simulating different user preferences, and selects the most expressive source task as the canonical task.

We first perform simulation experiments to evaluate the accuracy of predicting user actions in actual tasks of increasing sizes based on preferences learned from canonical tasks selected using our proposed metric. We then analyze how the transfer accuracy is affected by the size of the canonical task and the task-agnostic features. Lastly, we evaluate our proposed approach with real users in an online study where

MTurk participants play an assembly game. Overall, our results show that the canonical tasks selected using our proposed expressiveness metric lead to higher prediction accuracy in actual tasks as compared to randomly selected source tasks.

5.2 Related Work

The problem of transfer learning can be broken down into the following steps [101]; (i) selecting an appropriate source task to transfer from, (ii) understanding how the source and target tasks are related, and (iii) effectively transferring the learned model to the target task. Most of the prior work in this domain has relied on a human expert to select the source task and provide a mapping to the target task, and largely focused on using the policies learned in the source task as priors to speed up learning in the target task [100, 19, 27].

The source task is usually a simpler version of the target task. For example, in a mountain car scenario [19], a 2D version of the task is chosen as the source task for a 3D target task. The action $left$ in a 2D source task is mapped to $left$ and up in the 3D target task, while $right$ in 2D is mapped to $right$ and $down$ in 3D. In a robot soccer task [100], the weights for the new states and actions in a 4-vs-3 target task are initialized by duplicating the weights learned for similar states and actions in a 3-vs-2 source task. Similarly, preferences of a simulated human from a (source) block stacking task with 2 red and 3 blue blocks are used to expedite learning in a target task with 1 extra red block [71].

More generally, the source and target tasks can differ along various dimensions like the objectives, states, actions, constraints, or dynamics of the two tasks. They may have the same states and actions but differ only on the reward function [23] or have the same reward structure but different state and action spaces [10]. Recent work [34] has also provided a taxonomy to categorize the differences in the states of the source and target tasks based on how they affect transfer learning. In ascending order of abstraction, the tasks can differ in their feature values, feature dimensions, reward functions, and action policies.

While there is some work on learning a mapping between the states [13] and objects [35] in the source and target tasks, the problem of selecting an appropriate source task without human guidance is largely unexplored. Prior work suggests that a good source task can be selected from a set of candidate tasks by measuring their similarity with the target task [25]. However, the proposed similarity metrics require computing the performance of the learned policy in the target task or computing the number of states where the rewards or values of the two tasks are similar. The similarity between tasks can also be computed based on the likelihood of observing samples from the target task in the candidate source tasks [58]. However, this approach assumes that the tasks have similar state and action spaces and the robot has access to data samples in the target task.

In this work, we assume that the source and target tasks can have different states and actions, but share the same reward function represented by a common set of task-agnostic features. Unlike prior work, our goal is to select a good source task without having access to user demonstrations and without performing cost evaluations on the target task. It is also important to note that most of the prior research in selecting source tasks has focused on transfer learning for simulated agents, whereas our focus is on transfer learning of human preferences.

5.3 Problem

Task description. Following the previous chapters, we define each task as a Markov Decision Process (MDP) defined by the tuple $M := (S, A, T, R, S_{end})$; where S is a finite set of states in the task, A is a finite set of discrete user actions, $T(s_{t+1}|s_t, a_t)$ is the transition probability function, $S_{end} \subset S$ is the set of terminal states, and $R(s_{t+1})$ is the user's reward function. We assume that S, A, T, and S_{end} are known for a given task, while R captures the unknown user preference.

Preference transfer. The goal of transfer learning is to estimate the user policy $\hat{\pi}_X$ in a complex target task M_X by transferring the rewards R learned from demonstrations Ξ_C provided by the user in a

simpler source task M_C. To enable this transfer, as in our prior work [73], we represent the user rewards in both tasks as a weighted sum of the same task-agnostic features ϕ, where the feature weights w capture how users prioritize each dimension of the task-agnostic feature space $\Phi \in \mathcal{R}^d$.

$$R(s) = w^T \phi(s) \quad \forall s \in \{S_C, S_X\} \tag{5.1}$$

Given the user demonstrations Ξ_C in a source task, we use maximum-entropy IRL [121, 110] to learn the feature weights $\hat{w}$. We assume that users maximize their long-term expected reward on the task. Thus, we perform value iteration [12] to compute the user policy $\hat{\pi}_X$ based on $R(s) = \hat{w}^T \phi(s)$ for $s \in S_X$.

Goal. Assuming that the robot knows the task-agnostic features ϕ for representing user preferences in a target task M_X, we want to find a *canonical task* $M_C^\star$, such that the policy $\hat{\pi}_X$ computed based on the transferred weights $\hat{w}$ closely matches the users' actual policy π_X.

5.4 Canonical Task Selection

User preferences learned over task-agnostic features can be effectively transferred from source to target tasks when: (i) the task-agnostic features are sufficient for representing all the different user preferences in the target task, and (ii) the feature weights for different users can be accurately learned from their demonstrations in the canonical task. Since we assume that the robot knows the relevant task-agnostic features ϕ for modeling user preferences, we focus on selecting a canonical source task $M_C^\star$ such that we can accurately learn the weights (i.e., $\hat{w} = w$) for users with different preferences $w \in W$.

5.4.1 Evaluating the expressiveness of source tasks

For estimating the policy of any given user in the target task, the robot must learn their weights w for the task-agnostic features ϕ from their demonstrations Ξ_C in the canonical task. We assume that the user

preferences (i.e., weights) lie in a continuous space W and the user demonstrations are trajectories (i.e., sequence of state-action pairs) that are optimal with respect to their weights.

To accurately recover the weights for every user $w_i \in W$ based on their demonstrations $\Xi_{C,i}$, we would need the demonstrated trajectories to only be optimal for that user. In other words, we need the mapping from weights to demonstrations to be injective, i.e., $f : W \mapsto \Xi_C$ such that if $w_i \neq w_j$, then $f(w_i) \neq f(w_j)$. If the same demonstrations are optimal for a number of different weights, we cannot discern the true weights that produced the demonstrations. However, since our space of user preferences is continuous, achieving such a one-to-one mapping would not be feasible as it would require the canonical task to allow for an infinite number of unique demonstrations.

This is typical in IRL problems where several different weights can produce the same demonstrations. Prior work has argued that the ambiguity in determining the true weights is caused by the representation as well as the experiment [5, 4]. For example, in our representation, every demonstration is optimal for weights $w = \vec{0}$. Moreover, scaling the weights will not change the resulting demonstrations (i.e., $w \equiv \alpha w$ for any $\alpha > 0$). On the other hand, experimental ambiguity is an artifact of the task structure, i.e., the states, actions, and transitions. For example, a task where the actions are constrained such that all users are forced to choose the same demonstration or a task where all demonstrations have the same reward.

Both tasks in the above example do not allow users to uniquely express their preferences. To mitigate the experimental ambiguity in recovering different user weights based on their demonstrations, we need the canonical task to be as expressive as possible. Based on our key insight, we define expressiveness $\mathcal{E}$ as the capacity of a task to allow users with different weights to have different optimal demonstrations. Thus, to evaluate the expressiveness of a given canonical task, we propose a metric for expressiveness $d_{\mathcal{E}}$ that is proportional to the number of unique demonstrations obtained by simulating W_s users uniformly sampled from W.

$$d_{\mathcal{E}} = \frac{|\{\Xi_s\}|}{|\{W_s\}|} \tag{5.2}$$

$d_{\mathcal{E}} = 1$ when each weight $w_i \in W_s$ corresponds to a different set of demonstrations and $d_{\mathcal{E}} < 1$ when there is more than one weight that leads to the same demonstrations. Thus, lower values of $d_{\mathcal{E}}$ correspond to lower expressiveness and vice versa.

There are several ways to measure whether two demonstrations are the same; based on their end state, based on their sequence of actions, states, or features, or based on their rewards. The correct choice for comparison depends on the approach used to learn the weights from demonstrations and transfer them to the target task [25]. In this work, we use maximum-entropy IRL, which learns the weights for the user's reward function by minimizing the difference in the expected cumulative feature counts $\bar{F}(\Xi_i)$ of the learner's policy and the user's demonstrations. Thus, we consider any two demonstrations to be different if they have different cumulative feature counts and revise the expressiveness metric.

$$\bar{F}(\Xi_i) = \frac{1}{|\Xi_i|} \sum_{\xi \in \Xi_i} \sum_{s \in \xi} \phi(s) \tag{5.3}$$

$$d_{\mathcal{E}} = \frac{|\{\bar{F}(\Xi_i) \mid \Xi_i \in \Xi_s\}|}{|\{W_s\}|} \tag{5.4}$$

While we can compute and discern the optimal demonstrations for simulated users, real users may not be able to distinguish between demonstrations that have similar, but not the same, cumulative feature counts. Thus, to make it easier for users to realize which demonstration is optimal for their weights, we would want the optimal demonstrations in the canonical task to be as different from each other as possible.

Algorithm 4 Selecting canonical tasks for transfer learning

Require: Task generator G, Weights W_s, Features ϕ

for $n = 2, \ldots, N$ **do**
  Initialize $\mathcal{M} = [\,]$
  for $k = 1, \ldots, K$ **do**
    Generate $M_C \leftarrow G(n, \phi)$
    Initialize $\Xi = \emptyset$
    for $w_i \in W_s$ **do**
      $\pi_{w_i} \leftarrow computePolicy(w_i, M_C)$
      Rollout Ξ_i by simulating π_{w_i} in M_C
      $\Xi = \Xi \cup \Xi_i$
    Evaluate $d_{\mathcal{E}}(\Xi, W_s, \phi)$ and $d_{\mathcal{D}}(\Xi, \phi)$
    Store $(M_C, d_{\mathcal{E}}, d_{\mathcal{D}})$ in $\mathcal{M}$
  Select $\mathcal{M}^\star = \arg\max_{M_C \in \mathcal{M}} \; d_{\mathcal{E}}$
  Select $M_C^\star = \arg\max_{M_C \in \mathcal{M}^\star} \; d_{\mathcal{D}}$

We measure this by computing the between cluster sum of squares (d_D), considering the mean cumulative feature counts for each weight $w_i \in W_s$ as a cluster.

$$d_{\mathcal{D}} = \frac{\sum_{\Xi_i \in \Xi_s} |\Xi_i| (\bar{F}(\Xi_i) - \bar{F}(\Xi_s))(\bar{F}(\Xi_i) - \bar{F}(\Xi_s))^T}{|\Xi_s| - 1} \tag{5.5}$$

Here, $\bar{F}(\Xi_s)$ is the mean cumulative feature count across all simulated user demonstrations. When selecting the best canonical task from a set of source tasks, we first pick the tasks with the highest score for expressiveness ($d_{\mathcal{E}}$). If there is only one such task, we select it as our canonical task. If there are multiple equally expressive tasks, we select the task with the highest score for $d_{\mathcal{D}}$ as the canonical task.

5.4.2 Selecting expressive canonical tasks

In order to select the canonical task using our proposed metric, we first need a set of candidate source tasks. The source tasks can be of different sizes and have different states, actions, and transitions. Here, the size of a task refers to the expected length of user demonstrations in that task. A larger canonical task can be more expressive but also require more human effort in providing demonstrations, while a shorter task will be easier to perform but may not be able to accurately recover a wide range of user weights.

Ideally, the canonical task should be less demanding than the actual task, but the amount of effort it should save will vary depending on the use case. To accommodate this, we select canonical tasks of different sizes, ranging from 2 to N, and allow the experimenter to have the flexibility of choosing the canonical task of a suitable size for their specific needs.

Our proposed approach for selecting the canonical tasks takes as input a random task generator G, the task-agnostic feature function ϕ, and a set of simulated user weights W_s (see Algorithm 4). For a given size $n \in N$ of the canonical task, we obtain K source tasks M_C from the task generator (step 4), evaluate their expressiveness (step 10) based on the simulated user demonstrations (step 8), and select the source task with the highest scores as the canonical task for that size (steps 12 and 13).

In addition to the metric for evaluating the expressiveness of source tasks, our approach has two key components: randomly generating the source tasks and sampling a set of user weights for evaluation. We provide details of the random task generator G and sampled user weights W_s used in our experiments in the following section.

5.5 Simulation Experiments

We first evaluate our proposed method for selecting canonical tasks with simulated users. Particularly, we measure the accuracy of predicting the (simulated) user actions in target tasks of different sizes, using preferences transferred from the selected canonical tasks. We compare different metrics for selecting the canonical task and analyze how the prediction accuracy varies with both the size of the canonical task and the dimensionality of the task-agnostic feature space.

5.5.1 Implementation

We now provide details of the random task generator G, the task-agnostic feature function ϕ, and the weights W_s used to evaluate the source tasks.

Algorithm 5 Randomly generating source tasks

Require: Task size n, Space of action properties P

1: Initialize empty actions $A = [a_1, \ldots, a_n]$
2: Randomly sample $p(a_i) \sim P$ for $a_i \in A$
3: **for** $i = 1, \ldots, n$ **do**
4: $\quad c(a_i) = [\,]$
5: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
6: $\quad\quad$ Randomly assign $c(a_i) \leftarrow a_j$
7: $s_{init} = [0]^n$ and $s_{end} = [1]^n$
8: $S = enumerateStates(s_{init}, s_{end}, A, c)$

5.5.1.1 Task generator

As we described in Section 5.3, each task is represented by its states S, actions A, transitions T, rewards R and terminal states S_{end}. While prior work has explored generating MDPs from hierarchical task networks (HTNs) [67], we choose to randomly generate the source tasks based on the desired task size (see Algorithm 5).

We first generate the set of actions in the task by assuming that each task action $a \in A$ is described by a d-dimensional vector of properties $p(a) = [p_1, \ldots, p_d]$. These properties correspond to the task-agnostic features such as the effort required to perform the action, user familiarity with the action, the tools required to perform the action, etc. Additionally, we assume that each action must be executed once for task completion. Thus, a task of size n would have n actions.

For a given task size, we randomly sample the properties of each action from a distribution P (step 2). P can be discrete or continuous. For our simulation experiments, we assume $P = [0., 0.5, 1.0]^2$ in Section 5.5.2 and $P = U[0, 1]^d$ in Section 5.5.3.

Next, we assign the sequencing constraints $c(a_i)$ for each action to determine the feasible transitions in the task. Here, $c(a_i)$ specifies the set of actions that must be performed before a_i. For example, if a part must be cleaned and polished before welding, $c(\text{weld\_part}) = \{\text{clean\_part}, \text{polish\_part}\}$. To ensure that the generated task can be successfully completed, we sequentially assign the constraints for each

generated action. The first action is unconstrained, and each subsequent action has a random chance of being sequentially constrained to any of the preceding actions (step 6).

Lastly, we represent the states of the task as one-hot encoded vectors of length n, where each dimension corresponds to an action in A and indicates whether that action has been performed. So, the state at the start of the task, when none of the actions have been performed, is represented by a vector of all 0s, while the terminal state, where all actions have been executed, is represented by all 1s. Starting from the initial state, we build a tree by executing the actions in A based on their constraints, to enumerate all the states in the task (step 8).

5.5.1.2 Feature function

We assume that user preferences for sequencing the task actions depend on the action properties and the feature function $\phi(s)$ that captures this dependency is provided to the robot.

In our simulation experiments, we model user preferences as frontloading actions that have high values for certain properties. For example, some users may prefer to start the task with actions that they are most familiar with, while other users may prefer to start with actions that require more effort. To capture this, we represent the features for each state in the task as the sum of properties of all the actions that have been executed in that state.

$$\phi(s) = \sum_{a_i \in A} s[a_i] \cdot p(a_i) \tag{5.6}$$

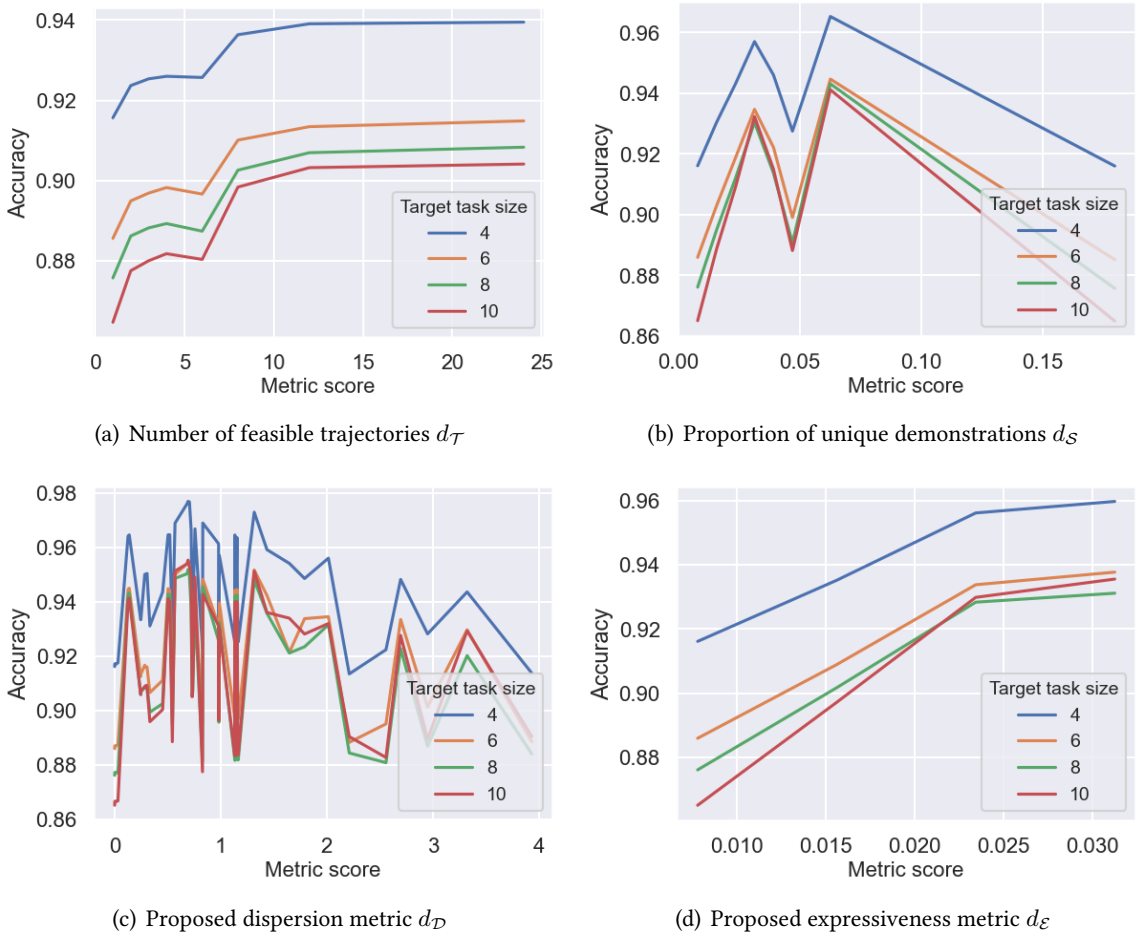
Consider the following example: Let a source task have two actions $p(a_1) = [0.9]$ and $p(a_2) = [0.1]$ where $p = [\text{familiarity}]$ and $c(a_1) = c(a_2) = \emptyset$. The starting and terminal states are $s_{init} = [0, 0]$ and $s_{end} = [1, 1]$, respectively. If the users prefer to start with actions that are more familiar, they will choose the sequence $\xi_1 : s_{init} \xrightarrow{a_1} [1, 0] \xrightarrow{a_2} s_{end}$ which has a cumulative feature count of $F(\xi_1) = [1.9]$, because it leads to a higher reward than the sequence $\xi_2 : s_{init} \xrightarrow{a_2} [0, 1] \xrightarrow{a_1} s_{end}$ which has $F(\xi_2) = [1.1]$.

5.5.1.3 Simulated users

Finally, we sample the set of weights W_s used to evaluate the task. To mitigate the representational ambiguity and create a diverse set of weights, we uniformly sample the weights from a unit d-sphere such that $||w|| = 1$. This prevents sampling weights that are scalar multiples of each other. Each dimension of the weight vector corresponds to a dimension of the task-agnostic feature space Φ.

In our simulation experiments, we limit the weights to positive values to solely model user preferences for frontloading actions based on their properties.

5.5.2 Comparison of metrics for canonical task selection



(a) Number of feasible trajectories $d_{\mathcal{T}}$

(b) Proportion of unique demonstrations $d_{\mathcal{S}}$

(c) Proposed dispersion metric $d_{\mathcal{D}}$

(d) Proposed expressiveness metric $d_{\mathcal{E}}$

Figure 5.1: Mean accuracy of predicting the simulated user actions in target tasks of different sizes based on their demonstrations in source tasks of varying metric scores.

We evaluate the effectiveness of our proposed metric in selecting canonical tasks that are suitable for the transfer learning of human preferences. Specifically, we measure the accuracy of predicting user actions in target tasks based on their demonstrations in source tasks, as the metric score increases. We also conduct ablations of our proposed metric to compare against the following baselines:

5.5.2.1 Number of feasible trajectories

To determine whether we need to simulate user demonstrations in the source task for measuring its expressiveness, we consider the number of feasible trajectories in the task ($d_{\mathcal{T}}$) as a baseline. While this baseline measures the different ways in which the task can be completed, it does not take into account that only a subset of these trajectories may be optimal for the users.

5.5.2.2 Proportion of unique demonstrations

For this baseline, we simulate user demonstrations in the source task and measure the proportion of demonstrations that are unique ($d_{\mathcal{S}}$), as in Equation 5.2. However, we determine the uniqueness of demonstrations based on their sequence of states instead of their cumulative feature counts as in Equation 5.4. This metric does not take into account that trajectories with different state sequences would be equivalent for the users if the trajectories have the same total reward.

5.5.2.3 Dispersion in feature counts

Lastly, we consider our metric $d_{\mathcal{D}}$, which measures the dispersion in feature counts of the simulated user demonstrations, as an independent baseline for measuring the expressiveness of tasks.

For each metric, we generate 256 source tasks of size 4 and consider a 2-dimensional task-agnostic feature space. We then simulate the demonstrations of 128 sampled users on each task and compute the metric scores. For each source task, we evaluate the accuracy of predicting the actions of 64 newly sampled users in 32 randomly generated target tasks based on their demonstrations in the source task.

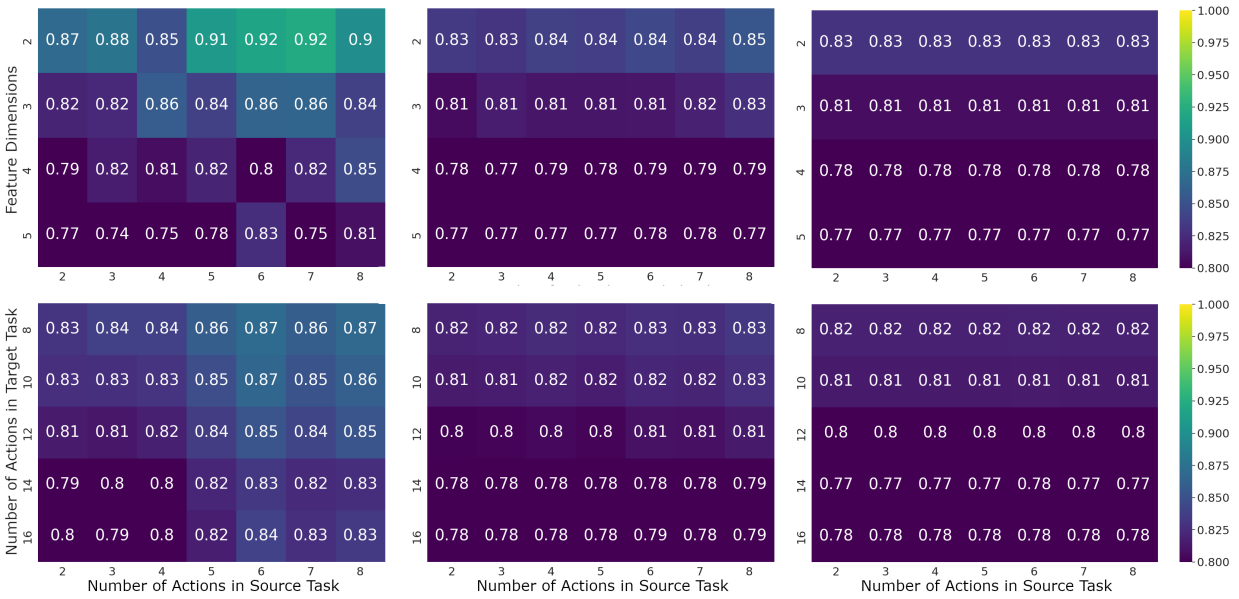
Table 5.1: Comparison of metrics for selection of canonical tasks.

$\lvert M_X \rvert$	Mean accuracy (Std. error)				ANOVA results	
	$d_{\mathcal{T}}$	$d_{\mathcal{S}}$	$d_{\mathcal{D}}$	$d_{\mathcal{E}}$	$F(3, 252)$	p
4	0.93 (0.002)	0.91 (0.003)	0.91 (0.003)	**0.96 (0.001)**	51.65	< 0.001
6	0.91 (0.001)	0.88 (0.003)	0.88 (0.002)	**0.93 (0.001)**	139.51	< 0.001
8	0.90 (0.001)	0.87 (0.003)	0.88 (0.002)	**0.93 (0.001)**	132.62	< 0.001
10	0.90 (0.001)	0.86 (0.005)	0.89 (0.004)	**0.93 (0.002)**	68.44	< 0.001

Here, accuracy is the percentage of actions in the user demonstrations that are correctly predicted based on the transferred weights. We do this for target tasks of different sizes, ranging from 4 to 10 actions.

Fig. 5.1(d) illustrates the correlation between prediction accuracy and the metric score when using our proposed metric. It indicates that tasks with higher scores on the metric result in higher prediction accuracy. Moreover, this behavior is consistent across target tasks of different sizes. We perform a two-way ANOVA to analyze the effect of the metric score and target task size on prediction accuracy. The test revealed a statistically significant interaction between the effects of the two independent variables ($F(9, 1008) = 4.178, p < 0.001$). Simple main effects analysis showed that both metric score and target task size had a statistically significant effect ($p < 0.001$) on the prediction accuracy.

Conversely, obtaining higher scores on the baselines does not necessarily indicate a greater accuracy of predictions. We compare the performance of our proposed metric to the baselines by measuring the prediction accuracy of the best-scoring tasks on each metric. For each size of the target task, we perform a one-way ANOVA (see Table 5.1). We found that using our proposed metric to select canonical tasks results in significantly higher prediction accuracy than the baseline metrics. Overall, these results attest to the quality of our proposed metric and its effectiveness in selecting canonical tasks.



(a) Best scoring tasks (b) Random tasks (c) Worst scoring tasks

Figure 5.2: Mean prediction accuracy for (a) the best scoring task on our proposed metric, (b) 32 randomly sampled source tasks, and (c) the worst scoring task on our proposed metric, in each grid cell, corresponding to the size of the canonical task, target task, and feature space.

5.5.3 Effect of canonical task size on prediction accuracy

We now test the performance of our proposed metric by varying the size of the canonical task, the dimensions of the task-agnostic feature space, and the size of the target tasks. We simulate 256 users, with 64 users each corresponding to feature spaces of 2, 3, 4, and 5 dimensions, and measure the accuracy of predicting their actions on 128 target tasks, consisting of 32 tasks each of sizes 10, 12, 14, and 16. The predictions are based on weights learned from user demonstrations in canonical tasks with sizes ranging from 2 to 8. To select the canonical tasks, we sample 512 source tasks and score them using our proposed metric by simulating 128 sampled users.

Fig. 5.2(a) (top) shows the prediction accuracy for the best-scoring tasks on our proposed metric. Each cell depicts the mean accuracy across all target tasks. We see that the prediction accuracy improves as we increase the size of the canonical task. Larger source tasks allow users to provide more distinct demonstrations, which makes them more expressive and hence achieve higher prediction accuracy. We also find

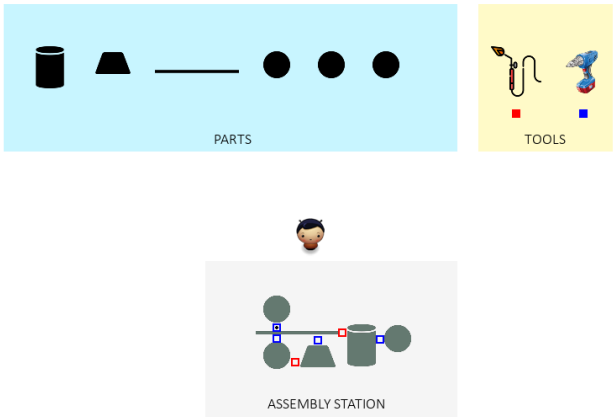
that we need larger canonical tasks to maintain performance as the feature dimensions increase. This is because high-dimensional feature spaces can encode a wider range of user preferences in the target tasks, thus requiring source tasks that are more expressive. We see a similar result for increasing sizes of target tasks in Fig. 5.2(a) (bottom). Here, each cell depicts the mean accuracy across all feature dimensions. Our results indicate that when the feature dimensions and target task sizes increase, we require larger canonical tasks for effectively transferring user preferences.

To further analyze the benefit of selecting tasks using our proposed metric, we compare the best-scoring tasks to the worst-scoring tasks on our proposed metric and to randomly selected source tasks. Each cell in Fig 5.2(b) illustrates the mean accuracy for 32 random source tasks. We select the worst-scoring tasks by minimizing our proposed metric in step 12 of Algorithm 4. Overall, the results show that canonical tasks selected using our metric are more useful for transfer learning of user preferences, achieving higher accuracy in predicting user actions in the target tasks.

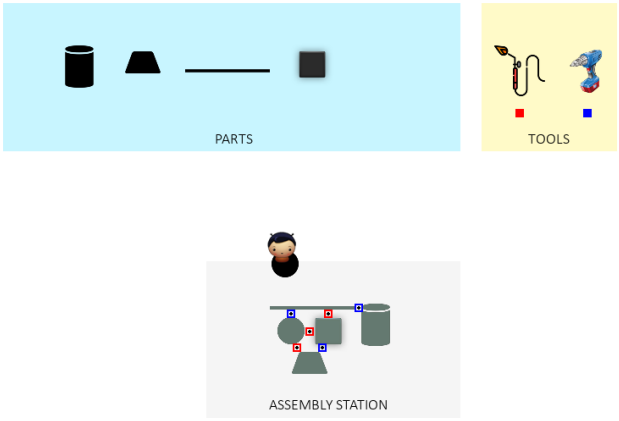
In this experiment, we make an interesting observation regarding the comparison between the best and worst scoring tasks. We found that the worst-scoring tasks tend to have more constrained actions, whereas the best-scoring tasks tend to have almost no ordering constraints. Moreover, the best-scoring tasks have unique actions with diverse properties, thus resulting in user demonstrations with distinct feature counts.

5.6 Online Assembly Study

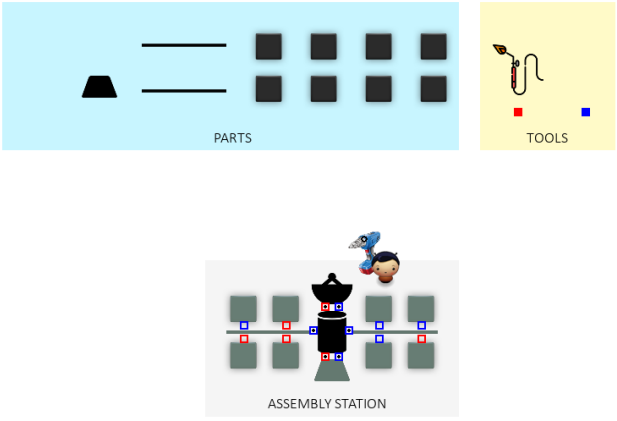
Finally, we test our proposed approach for selecting canonical tasks with real users in an online assembly game. Specifically, we compare the accuracy of predicting user actions in a target assembly task, based on their demonstrations in the best and worst scoring canonical tasks on our proposed expressiveness metric. Our hypothesis is that transferring human preferences from a more expressive canonical task will lead to higher prediction accuracy than a less expressive canonical task (**H1**).



(a) Worst scoring task



(b) Best scoring task



(c) Target task

Figure 5.3: Game environments for the three tasks in our online user study. For each task, the parts are initially placed in the blue section on the left and the tools are placed in the yellow region on the right, at the top of the screen. The grey shapes in the assembly station indicate where the parts must be placed, while the small red and blue squares indicate where the assembly operations must be performed. The black dots inside the action squares indicate that the action is not constrained and can be performed next.

5.6.1 Task description

The assembly game involves controlling a character to move parts from their initial positions to the corresponding locations in the assembly station and using the appropriate tools to connect the parts (see Fig. 5.3). The target task resembles a satellite assembly that consists of 14 actions with the only constraint that the horizontal rods must be connected to the satellite body before the square panels can be attached. The properties of each action specify the parts and tools required to perform that action. Based on a pilot study, we consider a 4-dimensional feature function, $\phi = [\text{same\_part}, \text{same\_tool}, \text{diff\_part}, \text{diff\_tool}]$, to model user preferences for minimizing (or maximizing) the number of times they have to change the tools and parts during the task.

We generate 256 source tasks of size 6 by sampling action properties from a discrete space of parts and tools and select the best-scoring task by simulating 128 sampled users. As a baseline, we select the worst-scoring task by minimizing our proposed metrics in steps 12 and 13 of Algorithm 4. Each subsequent action in the worst-scoring task is constrained by the previous action, forcing each user to provide the same demonstration. On the other hand, the actions in the best-scoring task do not have any sequencing constraints.

5.6.2 Study protocol

We recruited 22 participants from Amazon Mechanical Turk (AMT) and used Qualtrics to record their survey responses. Each participant played the game hosted on Trinket and performed all three tasks: the best-scoring canonical task, the worst-scoring source task, and the target assembly task. For each task, the participants were provided a practice round to understand the states, actions, and transitions in the task, and determine their preferred sequence. After the practice round, the participants were instructed to demonstrate their preferred sequence of actions for completing the task as fast as possible. At the end of the task, the participants returned to the survey to explain their preferences.

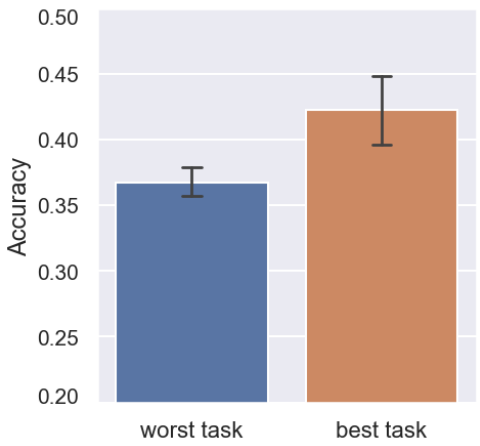


Figure 5.4: Mean accuracy of predicting user actions in the target task based on their demonstrations in the best and worst scoring tasks.

5.6.3 Offline evaluations

For each user, we use their demonstrations in the source tasks to learn the feature weights, compute their objective function and measure the accuracy of predicting their demonstration in the target task. We noticed that a fair number of MTurk participants provided sub-optimal demonstrations resulting in lower prediction accuracy. Yet, a two-tailed paired t-test showed a statistically significant difference ($t(21) = 2.09$, $p = 0.048$) in prediction accuracy of the best ($M = 0.42$, $SE = 0.025$) and worst ($M = 0.36$, $SE = 0.011$) scoring tasks (see Fog. 5.4). This supports hypothesis **H1** and validates the benefit of using our proposed approach for the selection of canonical tasks.

5.7 Conclusion

Our work proposes a novel metric for selecting source tasks that are suitable for transfer learning of human preferences. Although our metric is tailored to a specific IRL approach, we believe that our key insight of using expressive source tasks can be adapted for other methods. Our simulation experiments demonstrate that our proposed expressiveness metric can identify source tasks that result in better prediction accuracy. We also analyze the performance of our approach with varying sizes of source and target tasks and feature

dimensions. Finally, our online study shows that our method can also be beneficial for transferring the preferences of real users.

Even so, the performance of our proposed approach is dependent on the set of generated source tasks and the simulated user weights. In problems requiring larger source tasks and higher dimensional feature spaces, the best canonical tasks can be difficult to produce using a random task generator. Hence, future work can leverage techniques from quality diversity optimization [36] for efficiently searching the space of tasks. Moreover, evaluating tasks using a large number of sampled users can be computationally expensive. Future work can explore using methods for subset selection [59] to create a smaller but diverse set of users for measuring the expressiveness of source tasks.

Chapter 6

Conclusions

6.1 Summary

We have focused on the problem of proactively assisting users by predicting their actions in real-world assembly tasks. Accurately predicting user actions in an actual assembly required learning the individual preferences of users. We considered two scenarios where robots can efficiently learn the preferences of new users without obtaining their demonstrations in the actual task:

First, when the robot has access to demonstrations of other users, we can identify the dominant models of user preferences in the actual task and use them to quickly infer the preferences of new users. Specifically, we showed that in a complex task, like a bookcase assembly, users can have dominant preferences at different resolutions, and proposed a two-stage clustering and inference method for predicting user actions. Second, if the robot cannot obtain demonstrations in the actual task, we proposed learning their preferences in a canonical task and transferring them to the actual assembly. Our proposed system modeled user preferences as a weighted combination of the task-agnostic features shared by the two tasks. Through user studies, we showed that preferences learned in a manually designed canonical task could be used to predict user actions in a model-airplane assembly.

We extended our method of transferring user preferences to account for changing user preferences and demonstrated how the robot could proactively assist users in a collaborative human-robot assembly. We also showed how the canonical tasks can be automatically generated for a given assembly.

Overall, this dissertation makes the following contributions:

- While previous research has considered user preferences for how they want a robot to complete a task, our approach is different in that we model how the users prefer to perform the task themselves.
- We recognized that in complex assembly tasks, humans have dominant preferences at different levels of task abstraction and presented a two-stage approach for learning and inferring user preferences.
- Our research is among the pioneering efforts in transferring human preferences across tasks. We showed that user preferences learned over task-agnostic features in a short canonical task are useful for predicting their actions in a larger actual task. This can significantly reduce the human effort in demonstrating preferences and enable robots to efficiently adapt to users in the actual task.
- We also proposed an approach for automatically selecting canonical tasks for the transfer learning of human preferences to any given actual task. This can facilitate other researchers and end users in designing canonical tasks for their own applications.

6.2 Future Directions

Although we have evaluated leveraging the dominant and transferred human preferences in real-world tasks, we are excited about generalizing our work to different task configurations and human preference types, especially in domains other than assembly and manufacturing. Moreover, while we have predicted user actions for adapting the robot to the user's preference, the robot can also use the predictions to influence the user towards a different action sequence, in a mutual-adaptation formalism.

Our work has a number of applications; a personal robot can infer user preferences in activities of daily living like clearing a table, a search and rescue system can infer rescuer preferences for navigation and guide them towards favourable regions of the map. Such applications would require learning the relevant task-agnostic features in the specific domains. While deep learning techniques have shown promise in building latent task representations, the learned encoding is limited to the sensor data and prone to spurious correlations. Future work could employ feature learning techniques to learn the task-agnostic features from objective (e.g., feature traces) as well as subjective (e.g., surveys) human feedback.

We believe that we have only scratched the surface of the potential ways of learning human models from other users and other tasks. By reducing the human effort in training such models we hope that assistive robots working in close-proximity with humans can find applications in a lot more domains.

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*. 2004, p. 1.

[2] Nichola Abdo, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. "Organizing objects by predicting user preferences through collaborative filtering". In: *The International Journal of Robotics Research* 35.13 (2016), pp. 1587–1608.

[3] Sharath Chandra Akkaladevi, Matthias Plasch, Christian Eitzinger, Sriniwas Chowdhary Maddukuri, and Bernhard Rinner. "Towards learning to handle deviations using user preferences in a human robot collaboration scenario". In: *International Conference on Intelligent Human Computer Interaction*. Springer. 2016, pp. 3–14.

[4] Kareem Amin, Nan Jiang, and Satinder Singh. "Repeated inverse reinforcement learning". In: *Advances in neural information processing systems* 30 (2017).

[5] Kareem Amin and Satinder Singh. "Towards resolving unidentifiability in inverse reinforcement learning". In: *arXiv preprint arXiv:1601.06569* (2016).

[6] Panagiotis Antonellis, Christos Makris, and Nikos Tsirakis. "Algorithms for clustering clickstream data". In: *Information Processing Letters* 109.8 (2009), pp. 381–385.

[7] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

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

[9] Arindam Banerjee and Joydeep Ghosh. "Clickstream clustering using weighted longest common subsequences". In: *Proceedings of the web mining workshop at the 1st SIAM conference on data mining*. Vol. 143. 2001, p. 144.

[10] Bikramjit Banerjee and Peter Stone. "General Game Learning Using Knowledge Transfer." In: *IJCAI*. 2007, pp. 672–677.

[11] Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. “Fast optimal leaf ordering for hierarchical clustering”. In: *Bioinformatics* 17.suppl_1 (2001), S22–S29.

[12] Richard Bellman. “A Markovian decision process”. In: *Journal of mathematics and mechanics* 6.5 (1957), pp. 679–684.

[13] Yoshua Bengio. “Deep learning of representations for unsupervised and transfer learning”. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 17–36.

[14] Erdem Biyik and Dorsa Sadigh. “Batch active preference-based learning of reward functions”. In: *Conference on Robot Learning*. 2018, pp. 519–528.

[15] Erdem Bıyık, Nicolas Huynh, Mykel J Kochenderfer, and Dorsa Sadigh. “Active preference-based gaussian process regression for reward learning”. In: *Robotics: Science and Systems*. 2020.

[16] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. “Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences”. In: *The International Journal of Robotics Research* 41.1 (2022), pp. 45–67.

[17] Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. “Asking easy questions: A user-friendly approach to active reward learning”. In: *Proceedings of the 3rd Conference on Robot Learning*. 2019.

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

[19] Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. “Policy Transfer using Reward Shaping.” In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. 2015, pp. 181–188.

[20] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. “Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration”. In: *arXiv preprint arXiv:1702.08212* (2017).

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

[22] Tadeusz Caliński and Jerzy Harabasz. “A dendrite method for cluster analysis”. In: *Communications in Statistics - Theory and Methods* 3.1 (1974), pp. 1–27.

[23] Zhangjie Cao, Minae Kwon, and Dorsa Sadigh. “Transfer Reinforcement Learning across Homotopy Classes”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2706–2713.

[24] Marc G Carmichael, Richardo Khonasty, Stefano Aldini, and Dikai Liu. “Human preferences in using damping to manage singularities during physical human-robot collaboration”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 10184–10190.

[25] James L Carroll and Kevin Seppi. "Task similarity measures for transfer in reinforcement learning task libraries". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. IEEE. 2005, pp. 803–808.

[26] Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. "Trust-aware decision making for human-robot collaboration: Model learning and planning". In: *ACM Transactions on Human-Robot Interaction (THRI)* 9.2 (2020), pp. 1–23.

[27] Ignasi Clavera, David Held, and Pieter Abbeel. "Policy transfer via modularity and reward guiding". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1537–1544.

[28] Oscar Day and Taghi M Khoshgoftaar. "A survey on heterogeneous transfer learning". In: *Journal of Big Data* 4.1 (2017), pp. 1–42.

[29] David W Eccles and Gershon Tenenbaum. "Why an expert team is more than a team of experts: A social-cognitive conceptualization of team coordination and communication in sport". In: *Journal of Sport and Exercise Psychology* 26.4 (2004), pp. 542–560.

[30] Ali EL MEZOUARY, Brahim HMEDNA, and BAZ Omar. "An evaluation of learner clustering based on learning styles in MOOC course". In: *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*. IEEE. 2019, pp. 1–5.

[31] Ahmetcan Erdogan and Brenna D Argall. "Prediction of user preference over shared-control paradigms for a robotic wheelchair". In: *2017 International Conference on Rehabilitation Robotics (ICORR)*. IEEE. 2017, pp. 1106–1111.

[32] Ahmetcan Erdogan and Brenna D Argall. "The effect of robotic wheelchair control paradigm and interface on user performance, effort and preference: an experimental assessment". In: *Robotics and Autonomous Systems* 94 (2017), pp. 282–297.

[33] Leonard S Feldt, David J Woodruff, and Fathi A Salih. "Statistical inference for coefficient alpha". In: *Applied Psychological Measurement* 11.1 (1987), pp. 93–103.

[34] Tesca Fitzgerald, Ashok Goel, and Andrea Thomaz. "Abstraction in data-sparse task transfer". In: *Artificial Intelligence* 300 (2021), p. 103551.

[35] Tesca Fitzgerald, Ashok Goel, and Andrea Thomaz. "Human-guided object mapping for task transfer". In: *ACM Transactions on Human-Robot Interaction (THRI)* 7.2 (2018), pp. 1–24.

[36] Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. "Covariance matrix adaptation for the rapid illumination of behavior space". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 2020, pp. 94–102.

[37] Lisa R Fournier, Emily Coder, Clark Kogan, Nisha Raghunath, Ezana Taddese, and David A Rosenbaum. "Which task will we choose first? Precrastination and cognitive load in task ordering". In: *Attention, Perception, & Psychophysics* 81.2 (2019), pp. 489–503.

[38] Barbara Furletti, Lorenzo Gabrielli, Chiara Renso, and Salvatore Rinzivillo. “Identifying users profiles from mobile calls habits”. In: *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*. ACM. 2012, pp. 17–24.

[39] Alborz Geramifard, Finale Doshi, Josh Redding, Nicholas Roy, and Jonathan P How. “Online discovery of feature dependencies”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, pp. 881–888.

[40] Matthew Gombolay, Anna Bair, Cindy Huang, and Julie Shah. “Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences”. In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 597–617.

[41] Matthew Craig Gombolay, Cindy Huang, and Julie Shah. “Coordination of human-robot teaming with human task preferences”. In: *2015 AAAI Fall Symposium Series*. 2015.

[42] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. “Policy shaping: Integrating human feedback with reinforcement learning”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 2625–2633.

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

[44] Yue Guo, Rohit Jena, Dana Hughes, Michael Lewis, and Katia Sycara. “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. 2021, pp. 784–791.

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

[46] Constanze Hesse, Karina Kangur, and Amelia R Hunt. “Decision making in slow and rapid reaching: Sacrificing success to minimize effort”. In: *Cognition* 205 (2020), p. 104426.

[47] Bryan Higgs and Montasir Abbas. “A two-step segmentation algorithm for behavioral clustering of naturalistic driving styles”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE. 2013, pp. 857–862.

[48] Guy Hoffman. “Evaluating fluency in human–robot collaboration”. In: *IEEE Transactions on Human-Machine Systems* 49.3 (2019), pp. 209–218.

[49] Guy Hoffman and Cynthia Breazeal. “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*. 2007, pp. 1–8.

[50] Christoffer Holmgård, Julian Togelius, and Georgios N Yannakakis. “Decision making styles as deviation from rational action: A super mario case study”. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 9. 1. 2013, pp. 142–148.

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

[52] Tariq Iqbal, Samantha Rack, and Laurel D Riek. “Movement coordination in human–robot teams: a dynamical systems approach”. In: *IEEE Transactions on Robotics* 32.4 (2016), pp. 909–919.

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

[54] Fumiaki Iwane, Manu Srinath Halvagal, Iñaki Iturrate, Iason Batzianoulis, Ricardo Chavarriaga, Aude Billard, and José del R Millán. “Inferring subjective preferences on robot trajectories using EEG signals”. In: *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE. 2019, pp. 255–258.

[55] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. “Learning preferences for manipulation tasks from online coactive feedback”. In: *The International Journal of Robotics Research* 34.10 (2015), pp. 1296–1313.

[56] Mingxuan Jing, Xiaojian Ma, Wenbing Huang, Fuchun Sun, and Huaping Liu. “Task transfer by preference-based cost learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 2471–2478.

[57] Przemyslaw A Lasota and Julie A Shah. “Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration”. In: *Human factors* 57.1 (2015), pp. 21–33.

[58] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. “Transfer of samples in batch reinforcement learning”. In: *Proceedings of the 25th International Conference on Machine learning*. 2008, pp. 544–551.

[59] Chengtao Li, Stefanie Jegelka, and Suvrit Sra. “Efficient sampling for k-determinantal point processes”. In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 1328–1337.

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

[61] Changchun Liu, Karla Conn, Nilanjan Sarkar, and Wendy Stone. “Online affect detection and robot behavior adaptation for intervention of children with autism”. In: *IEEE Transactions on Robotics* 24.4 (2008), pp. 883–896.

[62] Matthias Luber, Luciano Spinello, Jens Silva, and Kai O Arras. “Socially-aware robot navigation: A learning approach”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2012, pp. 902–907.

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

[64] Ana Madureira, Bruno Cunha, João Paulo Pereira, S Gomes, Ivo Pereira, Jorge M Santos, and Ajith Abraham. “Using personas for supporting user modeling on scheduling systems”. In: *2014 14th International Conference on Hybrid Intelligent Systems*. IEEE. 2014, pp. 279–284.

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

[66] J Maxwell Donelan, Rodger Kram, and Kuo Arthur D. “Mechanical and metabolic determinants of the preferred step width in human walking”. In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268.1480 (2001), pp. 1985–1992.

[67] Felipe Meneguzzi, Yuqing Tang, Katia Sycara, and Simon Parsons. “An approach to generate MDPs using HTN representations”. In: *Decision Making in Partially Observable, Uncertain Worlds: Exploring Insights from Multiple Communities*. 2011.

[68] Agathe Merceron and Kalina Yacef. “Clustering students to help evaluate learning”. In: *IFIP World Computer Congress, TC 3*. Springer. 2004, pp. 31–42.

[69] Daniel Müllner. “Modern hierarchical, agglomerative clustering algorithms”. In: *arXiv preprint arXiv:1109.2378* (2011).

[70] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. “Efficient behavior learning in human–robot collaboration”. In: *Autonomous Robots* 42.5 (2018), pp. 1103–1115.

[71] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. “Preference learning on the execution of collaborative human-robot tasks”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 879–885.

[72] Gonzalo Navarro. “A guided tour to approximate string matching”. In: *ACM Computing Surveys (CSUR)* 33.1 (2001), pp. 31–88.

[73] Heramb Nemlekar, Neel Dhanaraj, Angelos Guan, Satyandra K Gupta, and Stefanos Nikolaidis. “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*. 2023, pp. 575–583.

[74] Heramb Nemlekar, Runyu Guan, Guanyang Luo, Satyandra K Gupta, and Stefanos Nikolaidis. “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. 2022, pp. 1161–1167.

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

[76] Andrew Y Ng and Stuart Russell. “Algorithms for inverse reinforcement learning.” In: *Proceedings of the Seventeenth International Conference on Machine Learning*. 2000, pp. 663–670.

[77] Stefanos Nikolaidis, Anca Dragan, and Siddhartha Srinivasa. “Viewpoint-based legibility optimization”. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2016, pp. 271–278.

[78] Stefanos Nikolaidis, David Hsu, and Siddhartha Srinivasa. “Human-robot mutual adaptation in collaborative tasks: Models and experiments”. In: *The International Journal of Robotics Research* 36.5-7 (2017), pp. 618–634.

[79] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. “Efficient model learning from joint-action demonstrations for human-robot collaborative tasks”. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. 2015, pp. 189–196.

[80] Stefanos Nikolaidis and Julie Shah. “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)*. 2013.

[81] Stefanos Nikolaidis, Ryuichi Ueda, Akinobu Hayashi, and Tamio Arai. “Optimal camera placement considering mobile robot trajectory”. In: *2008 IEEE International Conference on Robotics and Biomimetics*. IEEE. 2009, pp. 1393–1396.

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

[83] Malayandi Palan, Gleb Shevchuk, Nicholas Charles Landolfi, and Dorsa Sadigh. “Learning reward functions by integrating human demonstrations and preferences”. In: *Robotics: Science and Systems*. 2019.

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

[85] Terry Peckham and Gord McCalla. “Mining student behavior patterns in reading comprehension tasks.” In: *International Educational Data Mining Society* (2012).

[86] Aniruddh G Puranic, Jyotirmoy V Deshmukh, and Stefanos Nikolaidis. “Learning from demonstrations using signal temporal logic”. In: (2021), pp. 2228–2242.

[87] Aniruddh G Puranic, Jyotirmoy V Deshmukh, and Stefanos Nikolaidis. “Learning From demonstrations using signal temporal logic in stochastic and continuous domains”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6250–6257.

[88] David V Pynadath, Ning Wang, Ericka Rovira, and Michael J Barnes. “Clustering behavior to recognize subjective beliefs in human-agent teams”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 1495–1503.

[89] Daniel Ramirez-Cano, Simon Colton, and Robin Baumgarten. “Player classification using a meta-clustering approach”. In: *Proceedings of the 3rd Annual International Conference Computer Games, Multimedia & Allied Technology*. 2010, pp. 297–304.

[90] Rajiv Ranganathan, Adenike Adewuyi, and Ferdinando A Mussa-Ivaldi. “Learning to be lazy: exploiting redundancy in a novel task to minimize movement-related effort”. In: *Journal of Neuroscience* 33.7 (2013).

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

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

[93] Ramón Rico, Miriam Sánchez-Manzanares, Francisco Gil, Carlos Maria Alcover, and Carmen Tabernero. “Coordination processes in work teams”. In: *Papeles del Psicólogo* 32.1 (2011), pp. 59–68.

[94] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. “Active preference-based learning of reward functions.” In: *Robotics: Science and Systems*. 2017.

[95] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. “Planning for autonomous cars that leverage effects on human actions.” In: *Robotics: Science and Systems*. Vol. 2. 2016.

[96] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. “Multimodal probabilistic model-based planning for human-robot interaction”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–9.

[97] Mariah L Schrum, Michael Johnson, Muyleng Ghuy, and Matthew C Gombolay. “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*. 2020, pp. 43–52.

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

[99] Malin Sundbom, Paolo Falcone, and Jonas Sjöberg. “Online driver behavior classification using probabilistic ARX models”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE. 2013, pp. 1107–1112.

[100] Matthew E Taylor and Peter Stone. “Behavior transfer for value-function-based reinforcement learning”. In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. 2005, pp. 53–59.

[101] Matthew E Taylor and Peter Stone. “Transfer learning for reinforcement learning domains: A survey.” In: *Journal of Machine Learning Research* 10.7 (2009).

[102] Ruck Thawonmas, Masayoshi Kurashige, and Kuan-Ta Chen. “Detection of landmarks for clustering of online-game players.” In: *International Journal of Virtual Reality* 6.3 (2007), pp. 11–16.

[103] Etienne Thuillier, Laurent Moalic, Sid Lamrous, and Alexandre Caminada. “Clustering weekly patterns of human mobility through mobile phone data”. In: *IEEE Transactions on Mobile Computing* (2017).

[104] Gustavo F Tondello, Alberto Mora, and Lennart E Nacke. “Elements of gameful design emerging from user preferences”. In: *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. ACM. 2017, pp. 129–142.

[105] Maegan Tucker, Myra Cheng, Ellen Novoseller, Richard Cheng, Yisong Yue, Joel W Burdick, and Aaron D Ames. “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. 2020, pp. 3423–3430.

[106] Manuela M Veloso, Peter Stone, and Michael Bowling. “Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer”. In: *Sensor Fusion and Decentralized Control in Robotic Systems II*. Vol. 3839. SPIE. 1999, pp. 134–141.

[107] Alan R Wagner. “Using cluster-based stereotyping to foster human-robot cooperation”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2012, pp. 1615–1622.

[108] Gang Wang, Xinyi Zhang, Shiliang Tang, Christo Wilson, Haitao Zheng, and Ben Y Zhao. “Clickstream user behavior models”. In: *ACM Transactions on the Web (TWEB)* 11.4 (2017), pp. 1–37.

[109] Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y Zhao. “Unsupervised clickstream clustering for user behavior analysis”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM. 2016, pp. 225–236.

[110] Weitian Wang, Rui Li, Yi Chen, Z Max Diekel, and Yunyi Jia. “Facilitating human–robot collaborative tasks by teaching-learning-collaboration from human demonstrations”. In: *IEEE Transactions on Automation Science and Engineering* 16.2 (2018), pp. 640–653.

[111] Wenshuo Wang, Junqiang Xi, Alexandre Chong, and Lin Li. “Driving style classification using a semisupervised support vector machine”. In: *IEEE Transactions on Human-Machine Systems* 47.5 (2017), pp. 650–660.

[112] Ronald Wilcox, Stefanos Nikolaidis, and Julie Shah. “Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing”. In: *Robotics: Science and Systems VIII* (2013), p. 441.

[113] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. “A survey of preference-based reinforcement learning methods”. In: *Journal of Machine Learning Research* 18.136 (2017), pp. 1–46.

[114] Gwen M Wittenbaum, Garold Stasser, and Carol J Merry. "Tacit coordination in anticipation of small group task completion". In: *Journal of Experimental Social Psychology* 32.2 (1996), pp. 129–152.

[115] Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. "ELAN: a professional framework for multimodality research". In: *5th International Conference on Language Resources and Evaluation (LREC 2006)*. 2006, pp. 1556–1559.

[116] Li Yujian and Liu Bo. "A normalized Levenshtein distance metric". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1091–1095.

[117] Andrea Maria Zanchettin, Andrea Casalino, Luigi Piroddi, and Paolo Rocco. "Prediction of human activity patterns for human–robot collaborative assembly tasks". In: *IEEE Transactions on Industrial Informatics* 15.7 (2018), pp. 3934–3942.

[118] Karl E Zelik and Arthur D Kuo. "Mechanical work as an indirect measure of subjective costs influencing human movement". In: *PLoS ONE* 7.2 (2012), e31143.

[119] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. "Large-scale parallel collaborative filtering for the netflix prize". In: *International Conference on Algorithmic Applications in Management*. Springer. 2008, pp. 337–348.

[120] Zuyuan Zhu and Huosheng Hu. "Robot learning from demonstration in robotic assembly: A survey". In: *Robotics* 7.2 (2018), p. 17.

[121] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. "Maximum entropy inverse reinforcement learning." In: *Proceedings of the 23rd National Conference on Artificial Intelligence*. Vol. 3. AAAI'08. Chicago, IL, 2008, pp. 1433–1438.

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