MARLUI: Multi-Agent Reinforcement Learning for Adaptive Point-and-Click UIs

THOMAS LANGERAK, ETH Zürich, Switzerland
SAMMY CHRISTEN, ETH Zürich, Switzerland
MERT ALBABA, ETH Zürich, Switzerland
CHRISTOPH GEBHARDT, ETH Zürich, Switzerland
CHRISTIAN HOLZ, ETH Zürich, Switzerland
OTMAR HILLIGES, ETH Zürich, Switzerland

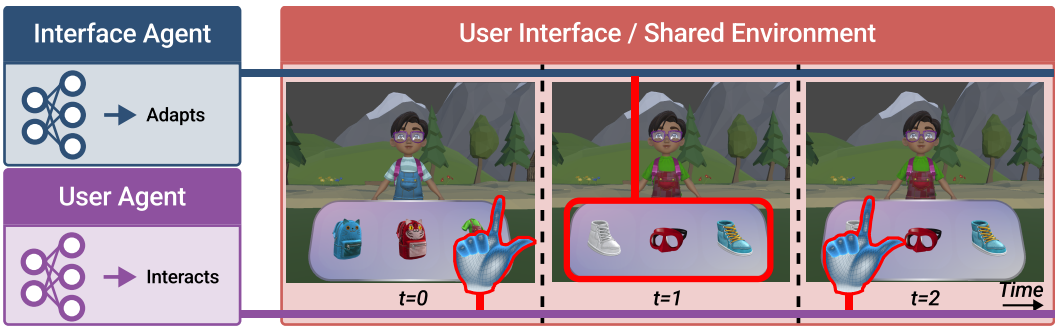

Fig. 1. We formulate the online adaptation of point-and-click interfaces as a multi-agent reinforcement learning problem. Our approach comprises a user- and interface agent. The user agent interacts with an application in order to reach a goal and the interface agent learns to assist. In this example, the user agent interacts with a toolbar, while the interface agent, in a turn-based manner, assigns relevant items for the user agent. After training, the interface agent can effectively assist real users in the same tasks.

As the number of selectable items increases, point-and-click interfaces rapidly become complex, leading to a decrease in usability. Adaptive user interfaces can reduce this complexity by automatically adjusting an interface to only display the most relevant items. A core challenge for developing adaptive interfaces is to infer user intent and chose adaptations accordingly. Current methods rely on tediously hand-crafted rules or carefully collected user data. Furthermore, heuristics need to be recrafted and data regathered for every new task and interface. To address this issue, we formulate interface adaptation as a multi-agent reinforcement learning problem. Our approach learns adaptation policies without relying on heuristics or real user data, facilitating the development of adaptive interfaces across various tasks with minimal adjustments needed. In our formulation, a user agent mimics a real user and learns to interact with an interface via point-and-click

Authors' addresses: Thomas Langerak, ETH Zürich, Department of Computer Science, Switzerland, thomas.langerak@inf.ethz.ch; Sammy Christen, ETH Zürich, Department of Computer Science, Switzerland, sammy.christen@inf.ethz.ch; Mert Albaba, ETH Zürich, Department of Computer Science, Switzerland, mert.albaba@inf.ethz.ch; Christoph Gebhardt, ETH Zürich, Department of Computer Science, Switzerland, christoph.gebhardt@inf.ethz.ch; Christian Holz, ETH Zürich, Department of Computer Science, Switzerland, christian.holz@inf.ethz.ch; Otmar Hilliges, ETH Zürich, Department of Computer Science, Switzerland, otmar.hilliges@inf.ethz.ch.

2573-0142/2024/6-ART253 $15.00
https://doi.org/10.1145/3661147

actions. Simultaneously, an interface agent learns interface adaptations, to maximize the user agent's efficiency, by observing the user agent's behavior. For our evaluation, we substituted the simulated user agent with actual users. Our study involved twelve participants and concentrated on automatic toolbar item assignment. The results show that the policies we developed in simulation effectively apply to real users. These users were able to complete tasks with fewer actions and in similar times compared to methods trained with real data. Additionally, we demonstrated our method's efficiency and generalizability across four different interfaces and tasks.

CCS Concepts: • **Human-centered computing → Graphical user interfaces**; **User models**; **HCI theory, concepts and models**.

Additional Key Words and Phrases: Multi-Agent Reinforcement Learning, Adaptive User Interfaces, Intelligent User Interfaces

ACM Reference Format:
Thomas Langerak, Sammy Christen, Mert Albaba, Christoph Gebhardt, Christian Holz, and Otmar Hilliges. 2024. MARLUI: Multi-Agent Reinforcement Learning for Adaptive Point-and-Click UIs. *Proc. ACM Hum.-Comput. Interact.* 8, EICS, Article 253 (June 2024), 28 pages. https://doi.org/10.1145/3661147

1 INTRODUCTION

Point-and-click interfaces form the core interaction paradigm in modern Human-Computer Interaction (HCI) [54, 60, 89]. These interactions include clicking toolbar items, navigating hierarchical menus, and selecting objects—from UIs on traditional 2D desktops to emerging spatial UIs in immersive Mixed Reality. While individual point-and-click interaction is simple, complexity increases with the number of available interactive items. The presence of more items requires users to process more information and, thus, to evaluate more options, both of which lead to an increased cognitive load. This additional demand for cognitive effort can diminish the user's experience and unnecessarily extend the time needed to complete tasks.

Adaptive User Interfaces (AUIs) seek to mitigate this complexity by dynamically adjusting the interface presented to the user to display the items that are most relevant to the user's current task, while omitting less relevant information. This simplifies visual information processing and eliminates the user's need for navigating through complex interfaces such as hierarchical menus that require multiple actions to eventually select one item.

However, developing effective AUIs poses numerous challenges: AUIs must (1) access or infer users' intentions by observing their behavior and input, and they must (2) adapt the user interface based on these inferred intentions [84]. To map user input to adaptations, prior research has predominantly used machine learning (ML) techniques [34, 99, 100, 108], or heuristics [8, 102, 104]. However, the reliance of these methods on manually collected data or carefully hand-crafted rules is a significant limitation, as each new user interface or task requires new data collection and updated or redesigned adaptations, even when the interaction paradigm itself remains entirely unchanged.

The example in Fig. 1 illustrates these challenges for the simple scenario of customizing a game character through a toolbar interface. The toolbar has a limited number of slots to select from, whereas the set of clothing items is much larger, such that not all of them can be assigned to the available slots. To facilitate efficient interaction, an AUI needs to assign those clothing items to the available slots that the user is most likely to choose given their previous selections and the likelihood of intended final outfits. Now consider a VR game in which a user builds a castle from different blocks (Fig. 9). Here, the AUI recommends the most likely block to be selected for each operation, taking into account the user's past choices and the probabilities of final castle designs. Both UIs follow the point-and-click paradigm and the AUI's task is similar—yet machine learning or heuristic approaches would require separate collection of data as well as rule design for either

task. Considering this requirement for such simple tasks already, developing more complex AUIs can quickly become time-consuming and costly.

In this paper, we propose MARLUI, a proof-of-concept framework for easy development of AUIs across various point-and-click UIs. We frame the point-and-click interaction paradigm in a Multi-Agent Reinforcement Learning (MARL) framework: An interface agent learns to adapt a UI by selecting the most relevant subset of items at each interaction step. The interface agent observes the user agent and minimizes minimizes its task completion time, while the user agent learns human-like point-and-click behavior to interact with the UI and, thus, train the interface agent. Where previous approaches required real user data to teach an interface agent, our approach instead relies on a human-like user agent. The user agent and the interface agent jointly learn by exploring the interface through trial-and-error. To benefit real users during interaction, our interface agent transfers the learned adaptation strategy to actual scenarios where it selects the most relevant items after a real user's click or selection. When switching between different tasks within the MARLUI framework, such as from dressing a game character to constructing a block castle, the user agent and the interface agent require no to minimal modification and only need training on the respective user interface. Thus, our framework eliminates the need for developers to gather or create task- and interface-specific data or heuristics, thereby streamlining the AUI development process. However, there is space in future work for user personalization, extension to more complex interfaces, adapt to changing goals, and different tasks

To model the point-and-click paradigm as a MARL problem, we propose a simulated user agent to learn interaction with a UI for task completion. We model the user agent hierarchically by decomposing decision-making, such as selecting a toolbar item, from motor control actions, such as moving the cursor to a desired menu slot. Targeting human-like behavior in the user agent, we incorporate bounds on cognition and motor control, following Computational Rationality [83]. Concurrently, we train an interface agent. While the goal of the user agent is to reach a goal state, the interface agent aims to change the UI such that the user agent achieves its goal more efficiently (i.e., minimize the number of clicks). The interface agent operates as a single policy aimed at assisting the user agent, for instance, by correctly assigning items to a limited number of slots on a toolbar. The user agent and the interface agent operate in a turn-based manner: the user agent takes an action, and then the interface agent makes an adaptation. This cycle continues until the task is completed (i.e., the state of the UI aligns with user agent's goal). The user agent and interface agent learn by jointly exploring the UI through trial-and-error. This novel approach enables our data- and heuristic-free method.

To demonstrate the effectiveness of our framework across various tasks and interfaces, we explore five use cases. The first involves customizing a game character using a toolbar, where the interface agent assigns appropriate items to slots. The second case features an adaptive numeric keypad design. In the third task, users build a block tower while the interface agent supplies the necessary blocks. The fourth case deals with selecting objects that are initially out of reach, where the interface agent brings the needed object closer. The final task is a photo editing task in which the interface agent anticipates which part of a hierarchical menu the user wants to use and opens it ahead of time. While these use-cases all involve point-and-click tasks, they cover a broad range of common interface types, from heads-up displays (toolbar and keypad) to world-anchored interfaces (block tower and reaching task), which demonstrates the versatility of our approach in accommodating different tasks and interface designs.

We evaluate our framework in two stages to examine an interface agent's capability to learn policies that increase actual user performance. In in-silico studies on the character creation task, we find that our interface agent is capable of generalizing to unseen goals. We then evaluate MARLUI on the same task with real participants and compare our framework against several data-driven

baselines. We find that training the interface agent with our simulated user agent enables real humans to reduce the amount of actions necessary compared to previous data-driven frameworks.

In summary, we make three key contributions in this paper:

(1) the first attempt at a multi-agent Reinforcement Learning approach that does not rely on real user data or hand-crafted heuristics to adapt point-and-click user interfaces in real-time.
(2) a hierarchical, cognitively inspired user agent that learns to operate a user interface and enables an interface agent to learn the underlying structure of the task purely by observing the action of the user agent in the UI, and
(3) experimental results from our evaluations that establish the effectiveness of our approach as well as five different use-cases to demonstrate its general applicability for point-and-click tasks.

2 RELATED WORK

This paper proposes multi-agent RL as a framework for adaptive UIs. Our method features a user agent that models human interaction behavior and an interface agent that adapts the UI to support the user agent. Most related to our work is research methods for adaptive UIs, and on computational user modeling.

2.1 Methods for Adaptive UIs

Adaptive UIs aim to automatically and intelligently adapt a UI to guide the user toward completing their task. By dynamically adjusting elements such as layout, content, and functionality, they help the user navigate complex UIs more efficiently. UI adaptation can either be offline, to computationally design an interface, or online, to adapt the UI according to users' goals. We will focus on online adaptive UIs and refer readers to two survey papers [80, 82] for an overview of computational UI design.

2.1.1 Heuristics, Bayesian Networks & Combinatorial Optimization. In early works, heuristic- or knowledge-based approaches are used to evaluate [81] or adapt the UI [8, 102, 104]. Similarly, multi-agent systems employ rule-based and message-passing approaches [90, 91, 111]. Another popular technique for AUIs is domain-expert-designed Bayesian networks [5, 44]. More recently, combinatorial optimization was used to adapt interfaces dynamically [15, 67, 85]. The downside of these approaches is that experts need to specify user goals using complex rule-based systems or mathematical formulations. Creating them comprehensively and accurately requires developers to foresee all possible user states, which is tedious and requires expert knowledge. Commonly, these approaches also get into conflict when multiple rules or objectives apply. This conflict often results in unintuitive adaptations. Finally, these methods need to be manually recrafted for every change in task or interface. In contrast, our method only requires the layout of the UI and generalizes without manual intervention. From its representation as an RL environment, we learn policies that meaningfully adapt the UI and realistically reproduce user behavior.

2.1.2 Supervised Learning. Leveraging machine learning can overcome the limitations of heuristic-, network-, and optimization-based systems by learning appropriate UI adaptions from user data. Traditional machine learning approaches commonly learn a mapping from user input to UI adaptation. Algorithms like nearest neighbor [62, 73], Naïve Bayes [25, 74], perceptron [99, 100], support vector machines [4], or random forests [75, 87] are used and models are learned offline [4] and online [99]. Due to the problem setting, these approaches require users' input to be highly predictive of the most appropriate adaptation. Furthermore, it restricts the methods to work in use cases where myopic planning is sufficient, i.e., a single UI adaptation leads users to their goal. In contrast, our

method considers multiple goals when selecting an adaptation and can lead users to their goal using sequences of adaptations.

More recent work overcomes the limitations stemming from simple input-to-adaptation mapping by following a two-step approach. They (1) infer users' intention based on observations and (2) choose an appropriate adaptation based on the inferred intent [84]. Such work uses neural networks, and user intention is modeled either explicitly [57, 103] or as a low-dimensional latent representation [92]. Furthermore, mixed-initiative settings are used to learn both a user model and an adaptation model [76]. However, these approaches are still highly dependent on training data, which may not even be available for emerging technologies. In contrast, our method does not depend on pre-collected user data *and* can learn supportive policies just by observing simulated user behavior.

2.1.3 Bandits & Bayesian Optimization. Bandit systems are a probabilistic approach often used in recommender systems [40]. In a multi-armed bandit setting, each adaptation is modeled as an arm with a probability distribution describing the expected reward. The Bayes theorem updates the expectation, given a new observation and prior data. Related work leverages this approach for AUIs [53, 56, 69]. Bayesian optimization is a sample-efficient global optimization method that finds optimal solutions in multi-dimensional spaces by probing a black box function [97]. In the case of AUIs, it is used to find optimal UI adaptations by sampling users' preferences [58, 59]. Both approaches trade off exploration and exploitation when searching for appropriate adaptations (i.e., exploration finds entirely new solutions, and exploitation improves existing solutions), rendering them suitable approaches to the AUI problem.

However, such methods are not able to plan adaptations over a sequence of interaction steps, i.e., they plan myopic strategies. In addition, these approaches need to sample user feedback to learn or optimize for meaningful adaptations and, hence, also rely on high-fidelity user data. In contrast, our method can plan adaptations over a sequence of interaction steps learned from realistic, simulated user data.

2.1.4 Reinforcement Learning. Reinforcement learning is a natural approach to solving the AUI problem, as its underlying decision-making formalism implicitly captures the closed-loop iterative nature of HCI [46]. It is a generalization of bandits and learns policies for longer horizons, where current actions can influence future states. This generalization enables selecting UI adaptations according to user goals that require multiple interaction steps. Its capability makes RL a powerful approach for AUIs with applications in dialog systems [31, 105], crowdsourcing [21, 47], sequential recommendations [13, 66, 68], information filtering [96], personalization[26, 101, 116], and mixed reality [34, 35]. Similar to our work is a model-based RL method that optimizes menu adaptations [108] or complements the reward function with human feedback [33].

Current RL methods sample predictive models [31, 47, 108] or logged user traces [34]. However, these predictive models and offline traces represent user interactions with non-adaptive interfaces. Introducing an adaptive interface will change user behavior; so-called co-adaptation [42, 72]. Hence, it is unclear if the learned model can choose meaningful adaptations when user behavior changes significantly due to the model's introduction. In contrast, our user agent learns to interact with the adapted UI; hence, our interface agent learns on behavioral traces from the adapted setting.

2.1.5 Multi-Agent Reinforcement Learning. MARL is a generalization of RL in which multiple agents act, competitively or cooperatively, in a shared environment [115]. Multi-agent systems are common in games [3, 49], robotics [48, 79], or modeling of social dilemmas [63, 114]. MARL is challenging since multiple agents change their behavior as training progresses, making the learning problem non-stationary. Common techniques to address this issue is via implicit [107] or

explicit [29] communication, centralized critic functions [71, 113], or curricula [70, 109]. We take inspiration from the latter and use a curriculum during the training of our agents.

Closest to our work is [22], which proposes a multi-agent system that maps 2D interface trajectories to actions for navigating 3D virtual environments. A user agent learns interactions on a 2D interface. A decoder that is trained on real user data maps the user agent's actions to 2D touch gestures. A second agent then translates these 2D touch gestures into 3D operations. In their setting, the interface agent does not observe the environment itself but receives the actions of the user agent as its state. Our work extends their setting to the case where the user agent and the interface agent observe and manipulate the same UI. Furthermore, we do not rely on real world user data.

2.2 Computational User Modelling

To optimize and adapt interfaces, a notion of what is a good interface given a context is necessary. Computational user models predict performance and are therefore essential for UI optimization [84]. Early work relies on heuristics [1, 9–11, 55] and on simple mathematical models [28, 43]. More recent work extends these models and, for instance, predicts the operating time for a linear menu [20], gaze patterns [93], or cognitive load [24, 32].

Recently, reinforcement learning gained popularity within the research area of computational user models. This popularity is due to its neurological plausibility [6, 30], allowing it to serve as a model of human cognitive functioning. The underlying assumption of RL in HCI is that users behave rationally within their bounded resources [38, 83]. There is evidence that humans use such strategy across domains, such as in causal reasoning [23] or perception [39]. In human-computer interaction, researchers have leveraged RL to automate the sequence of user actions in a KLM framework [64], to predict fatigue in volumetric movements [12] to automatically evaluate UIs [37, 50]. It was also used to explain search behavior in user interfaces [110] or menus [14] and as a model for multitasking [52]. Most similar to our work is research on hierarchical reinforcement learning for user modeling. Jokinen et al. [51] show that human-like typing can emerge with the help of Fitts' Law and a gaze model. Other works show that HRL can elicit human-like behavior in task interleaving [36] or touch interactions [51].

Inspired by this line of research, we design our user agent by decomposing high-level decision-making from motor control. We see a unique potential, in that both adaptive interfaces and user models are converging to the use of Reinforcement Learning, making our proposal of using Multi-Agent Reinforcement Learning for the joint training of a user and interface agent a natural continuation of recent work.

3 FRAMEWORK OVERVIEW

We define "adaptation" as the online alteration of an interface given current and past user input to support users in completing their task more efficiently. As such AUIs dynamically assign relevant interactive elements to an interface, thereby decreasing number of decisions and actions users need to take. MARLUI is a framework to create AUIs and can learn adaptive policies, that is neural networks that given an input predict suitable adaptations. MARLUI learns without needing data and only necessitates minimal adjustments to learn these policies for different AUIs. We specifically, assume the users to be Computational Rational [83] to make the problem tractable. In this section, we provide a high-level overview of the workings of our proposed framework. As shown in Fig. 2, MARLUI consists of three main components: a user interface as a shared environment (which gets adapted), a user agent (which learns to interact with the AUI), and an interface agent that learns an adaptation policy. Referencing the game character customization example illustrated in Fig. 1,

we first describe the function of each component individually, followed by an explanation of how MARLUI can support real users.

User Interface / Shared Environment. Our approach models UI adaptation as a MARL problem, where an interface agent learns from a simulated user agent's interactions with a user interface, i.e., the shared environment. In this shared environment, user- and interface agent take actions in a turn-based manner. The user agent performs a point-and-click maneuver, which changes the state of the UI, e.g., altering the clothing of the game character. The interface agent observes the action and the change to the interface and based on that selects a new subset of items to display, e.g., adjusting the clothing items displayed in the toolbar. In turn, the user agent observes this adaptation and takes a new action. This cycle continues till the task is complete, e.g., the desired clothing configuration is reached.

User Agent. The user agent aims to achieve its goal as fast as possible. In the context of game character creation, goals might involve selecting specific attributes for a character, such as the color of a shirt or backpack. The user agent can observe the visible parts of the interface, and knows its internal state. Based on these information, the user agent acts on the interface with point-and-click maneuvers, aiming to align the current state of the UI with its goal, e.g., aligning the current- with the desired clothing configuration. By constraining its motor behavior in a physiologically plausible manner, the user agent is bound to exhibit human-like behavior for pointing and selecting items. Through trial and error, the user agent will eventually learn a policy that allows it to realize this alignment.

Interface Agent. The interface agent adapts the UI in a turn-based manner to minimize the number of actions the user agent must perform to complete a task. Despite not knowing the user agent's specific goal, it learns the task structure (i.e., sequence of states) through observing the user agent's interactions with the UI. It can then learn to select the subset of items that are most relevant to the user agent at its current state, e.g., dynamically populating the toolbar with the cloting items that are most likely to be picked. Through trial and error, it can learn UI adaptation policies without relying on pre-collected user data or predefined heuristics.

Interaction with Real Users. By mimicking human-like point-and-click behavior through the user agent, the interface agent can learn to adapt UIs such that it also assist real users in accomplishing the same task. To apply the learned adaptive interface to real users, the setting is changed such that interface agent interacts with the actual users instead of the user agent. In a turn-based fashion, it selects the most relevant next items after each click or selection of the actual user according to what it has learned through interactions with the user agent.

4 BACKGROUND

We briefly introduce (multi-agent) reinforcement learning and its underlying decision processes. Specifically, we assume that users behave according to Computational Rationality (CR) [83]. Computational rationality is a framework that models human decision-making as a process of optimal actions with respect to the constraints of the environment, available information, and cognitive resources. It aims to understand how humans can achieve their goals by making the best possible use of their limited cognitive, perceptual, and motor capabilities to reason and make decisions.

The CR assumption allows us to formulate UI interaction as a bounded reward driven problem [83]. From this follows that it can be modeled as an MDP. Since neither the user agent nor the interface agent have access to the full state space (e.g., the interface agent does not know the user's goal state), it is a partially observable setting and intuitively formulated as a POMDP. a POMDP formulation is common in related work for similar types of problems [14, 34].

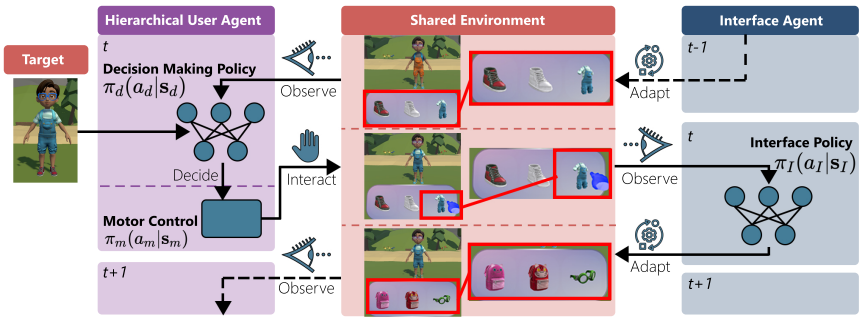

Fig. 2. Our interface agent and user agent act in the same environment. The user agent is modeled as a two-level hierarchy with a high-level decision-making policy π_d and a low-level motor control policy π_m. The user agent interacts with the UI. The high-level policy of the user agent observes that state of the environment (Eq. 1) and chooses a specific menu slot as target accordingly (Eq. 2). The lower level receives this action and computes a movement (Sec. 5.2.2). The interface agent policy π_I adapts the interface to assist the user agent in achieving its task more efficiently. It observes user actions in the UI (Eq. 6) and decides on adaptations. Note that the interface agent cannot access the goal of the user, making the problem partially observable.

4.1 Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDP) is a mathematical framework for single-agent decision-making in stochastic partially observable environments [2], which is a generalization over Markov Decision Processes [45]. A POMDP is a seven-tuple $(S, O, A, T, F, R, \gamma)$, where S is a set of states, O is set of observations and A is a set of actions. In POMDPs, the exact states ($s \in S$) of the evolving environment may or may not be captured fully. Therefore, observations ($o \in O$) represent the observable states, which may differ from the exact state. $T : S \times A \times S \rightarrow [0, 1]$ is a transition probability function, where $T(\mathbf{s}', \mathbf{a}, \mathbf{s})$ is the probability of the transition from state $\mathbf{s}'$ to $\mathbf{s}$ after taking action $\mathbf{a}$. Similarly, $F : S \times A \times O \rightarrow [0, 1]$ is an observation probability function, where $F(\mathbf{o}, \mathbf{a}, \mathbf{s}')$ is the probability of observing $\mathbf{o}$ while transitioning to $\mathbf{s}'$ after taking action $\mathbf{a}$. $R : S \times A \rightarrow \mathbb{R}$ is the reward function, discounted with factor γ.

4.2 Reinforcement Learning

Reinforcement Learning is a machine learning paradigm that rewards desired and penalizes undesired behavior. Generally, an agent observes an environment state and tries to take optimal actions to maximize a numerical reward signal. A key difference with supervised learning is that RL learns through exploration and exploitation rather than from an annotated dataset.

We follow the standard formulation of RL as an MDP [106], but use observations rather than states since the problem we are working on is partially observable. In our setting, the observation space is a subspace of the state space (the interface agent does not have access to the internal state of the user agent), and observations are deterministic: $F(\mathbf{o}, \mathbf{a}, \mathbf{s}') = 1$. The goal is to find an optimal policy $\pi : O \rightarrow A$, a mapping from observations to actions that maximizes the expected return: $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R_i(\mathbf{o}_t, \mathbf{a}_t)\right]$. Since both observation- and action spaces can be high-dimensional, neural networks are used for policy learning (i.e., we approximate the policy as π_θ, where θ are the learned parameters).

4.3 Multi-Agent Reinforcement Learning

Standard reinforcement learning formulations building upon MDPs or POMDPs assume a single policy. Stochastic games generalize MDPs for multiple policies [98]. When players do not have

perfect information about the environment, stochastic games become partial observable stochastic games. A partially observable stochastic game is defined as an eight-tuple $(N, \mathcal{S}, \mathcal{O}, \mathcal{A}, T, \mathcal{F}, \mathcal{R}, \gamma)$, where N is the number of policies. $\mathcal{S} = S_1 \times ... \times S_N$ is a finite set of state sets, and $\mathcal{O} = O_1 \times ... \times O_N$ is a finite set of observation sets, with subscripts indicating different policies. $\mathcal{A} = A_1 \times ... \times A_N$ is a finite set of action sets. T is the transition probability function. $\mathcal{F} = F_1 \times ... \times F_N$ defines a set of observation probability functions of different players. A set of reward functions is defined as $\mathcal{R} = R_1, ...R_N$. Furthermore, we define a set of policies as $\Pi = \pi_1, ...\pi_N$. Finally, γ is the discount factor.

All policies have their individual actions, states, observations, and rewards. In this paper, we optimize each policy individually, while the observations are influenced by each other's actions. We use model-free RL (for comparison to model-based RL, see [88]). This set of algorithms is used in an environment where the underlying dynamics $T(\mathbf{s}', \mathbf{a}, \mathbf{s})$ and $F(\mathbf{o}, \mathbf{a}, \mathbf{s}')$ are unknown, but it can be explored via trial-and-error. In the remainder of this paper, we use the terms state and observation interchangeably.

5 MULTI-AGENT REINFORCEMENT LEARNING FOR ADAPTIVE POINT-AND-CLICK UIS

We first present an outline the model of our user agent, consisting of a high- and low-level policy. Then we present the interface agent (Fig. 2).

5.1 General Task Description

We model tasks to be completed if the user agent achieves their desired goal. For game character creation, a goal can be the desired configuration of a character with a certain shirt (red, green, blue), and backpack (pink, red, blue). We represent the goal as a one-hot vector encoding $\mathbf{g}$ of these attributes. A one-hot vector can be denoted as $\mathbb{Z}_2^j$, where j is the number of items. For the previous example, $\mathbf{g}$ would be in $\mathbb{Z}_2^6$ as it possesses six distinct items.

Furthermore, the user agent can access an input observation denoted by $\mathbf{x}$, e.g., this can correspond to the current character configuration. The current input observation, $\mathbf{x}$, and the goal state $\mathbf{g}$ are identical in dimension and type.

The user agent interacts with the interface and attempts to match the input observation and goal state as fast as possible, such that $\mathbf{x} = \mathbf{g}$. Each interaction updates $\mathbf{x}$ accordingly, and a trial terminates once they are identical. In the character creation example, this would be the case if the shirt and backpack of the edited character are the same as the desired configuration. The interface agent makes online adaptations to the interface. It does *not* know the specific goal of a user. Instead, it needs to observe user interactions with the interface to learn the underlying task structure that will yield the optimal adaptations, e.g., the user likely wants to configure the shoes after configuring the backpack.

The interface agent learns to adapt the UI to the user agent by maximizing the same expected discounted reward. Specifically, the interface agent learns to infer optimal next adaptations over an infinite horizon by updating implicit probabilities of likely next actions of the user agent, given its current sequence of past actions. As such it does not learn an explicit or implicit goal probability distribution, but a distribution of the most likely next actions of the user agent. An example is to suggest a white and blue shoe to the user agent in the tool, as the user agent has not interacted with the category of shoes thus far (Fig. 1).

5.2 User Agent

The user agent interacts with an environment to achieve a certain goal (e.g., select the intended attributes of a character). The agent tries to accomplish this as fast and accurately as possible, hence it minimizes task completion time. Thus, the user agent first has to compare the goal state and input observation and then plan movements to reach the target. We model the user agent as a hierarchical agent with separate policies for a two-level hierarchy [61]. First, we introduce a high-level decision-making policy π_d that computes a target for the agent (high-level decision-making), we approximate visual cost with the help of existing literature [20]. Second, a WHo Model Fitts'-Law-based low-level motor policy π_m that decides on a strategy to reach this target. We now explain both policies in more detail.

5.2.1 High-level Decision-Making Policy. The high-level decision-making policy of the hierarchy is responsible to select the next target item in the interface. The overall goal of the policy is to complete a given task while being as fast as possible. Its actions are based on the current observation of the interface, the goal state, and the agent's current state. More specifically, the high-level state space S_d is defined as:

$$S_d = (\mathbf{p}, \mathbf{m}, \mathbf{x}, \mathbf{g}), \tag{1}$$

which comprises: i) the current position of the user agent's end-effector normalized by the size of the UI, $\mathbf{p} \in I^n$ (where n denotes the dimensions, e.g., 2D vs 3D), ii) an encoding of the assignment of each item $\mathbf{m} \in \mathbb{Z}_2^{n_i \times n_s}$, with n_i and n_s being the number of menu items and environment locations, respectively, iii) the current input state $\mathbf{x} \in \mathbb{Z}_2^{n_i}$, and iv) the goal state $\mathbf{g} \in \mathbb{Z}_2^{n_i}$. Here, I denotes the unit interval $[0, 1]$, and $\mathbb{Z}_2^n$ is the previously described set of integers. The item-location encoding m represents the current state of a UI. It can be used, for instance, to model item-to-slot assignments. The action space A_D is defined as:

$$A_d = \mathbf{t}, \tag{2}$$

which indicates the next target slot $\mathbf{t} \in \mathbb{N}_{n_s}$. The reward for the high-level decision-making policy consists of two weighted terms to trade-off between task completion accuracy and task completion time: i) how different the current input observation $\mathbf{x}$ is from the goal state $\mathbf{g}$, and ii) the time it takes to execute an action. Therefore, the high-level policy needs to learn how items correlate with the task goal as well as how to interact with any given interface. With this, we define the reward as follows:

$$R_d = \alpha \underbrace{\mathcal{E}_{gd}}_{i)} - (1-\alpha) \underbrace{(T_D + T_M)}_{ii)} + \mathbb{1}_{\text{success}}, \tag{3}$$

where $\mathcal{E}_{gd}$ is the difference between the input observation and the goal state, α a weight term, T_M the movement time as an output of the low-level policy, T_D the decision time, and $\mathbb{1}_{\text{success}}$ an indicator function that is 1 if the task has been successfully completed and 0 otherwise.

In addition to movement time, we also need to determine the decision time T_D. To this end, we are inspired by the SDP model [20]. This model interpolates between an approximated linear visual search-time component and the Hick-Hyman decision time [43] , both functions take into account the number of menu items and user parameters.

We define the difference $\mathcal{E}_{gd}$ between the input observation $\mathbf{x}$ and the goal state $\mathbf{g}$ as the number of mismatched attributes:

$$\mathcal{E}_{gd} = -\sum_{x \in \mathbf{g}, y \in \mathbf{x}} \frac{\mathbb{1}_{x \neq y}}{n_{attr}}, \tag{4}$$

where $\mathbb{1}$ is an indicator function that is 1 if $x \neq y$ and else 0, x and y are individual entries in the vectors $\mathbf{g}$ and $\mathbf{x}$ respectively, and n_{attr} is the number of attributes (e.g., shirt, backpack, and glasses).

5.2.2 Low-Level Motor Control Policy. The low-level motor control policy is a non-learned controller for the end-effector movement. In particular, given a target, it selects the parameters of an endpoint distribution (mean $\mu_\mathbf{p}$ and standard deviation $\sigma_\mathbf{p}$) . We set $\mu_\mathbf{p}$ to the center of the target. The target $\mathbf{t}$ is the action of the higher-level decision-making policy (A_D). Following empirical results [28], we set $\sigma_\mathbf{p}$ to 1/6th of a menu slot width to reach a hitrate of 96%.

Given the current position and the endpoint parameters (mean and standard deviation), we compute the predicted movement time using the Fitts' Law derived WHo Model [41].

$$T_M = \left(\frac{k}{(\sigma_\mathbf{p}/d_\mathbf{p} - y_0)^{1-\beta}} \right)^{1/\beta} + T_M^{(0)}, \tag{5}$$

where k and β are parameters that describe a group of users, $T_M^{(0)}$ is the minimal movement time, and y_0 is equal to the minimum standard deviation. The term $d_\mathbf{p}$ indicates the traveled distance from the current position to the new target position $\mu_\mathbf{p}$. We follow literature for the values of other parameters [41, 51]. We sample a new position from a normal distribution: $\mathbf{p} \sim \mathcal{N}\left(\mu_\mathbf{p}, \sigma_\mathbf{p}\right)$.

5.3 Interface Agent

The interface agent makes discrete changes to the UI to maximize the performance of the user agent. In our running example of character customization, it assigns items to a toolbar to simplify their selection for the user agent. Unlike the user agent, we model the interface agent as a flat RL policy. The state space S_I of the interface agent is defined as:

$$S_I = (\mathbf{p}, \mathbf{x}, \mathbf{m}, \mathbf{o}), \tag{6}$$

which includes: i) the position of the user $\mathbf{p} \in I^2$, ii) the input observation $\mathbf{x} \in \mathbb{Z}_2^{n_i}$, iii) the current state of the UI $\mathbf{m} \in \mathbb{Z}_2^{n_i \times n_s}$, and iv) a vector including the history of interface elements the user agent interacted with (commonly referred to as stacking). The action space $A_I \in \mathbb{Z}$ and its dimensionality is application-specific. The goal of the interface agent is to support the user agent. Therefore, the reward of the interface agent is directly coupled to the performance of the user agent. We define the reward of the interface agent to be the reward of the user agent's high-level policy:

$$R_I = R_D. \tag{7}$$

Note that the interface agent does *not* have access to the user agent's goal $\mathbf{g}$ or target $\mathbf{t}$. To accomplish its task, the interface agent needs to learn to help the user agent based on an implicit understanding of i) the objective of the user agent, and ii) the underlying task structure. Our setting allows the interface agent to gain this understanding solely by observing the changes in the interface as the result of the user agent's actions. This makes the problem more challenging but also more realistic.

The interface agent learns an implicit distribution of possible goals, and by observing the user agent it narrows down the distribution over goals. At every time step the interface agent suggests the items that are most likely needed, given the goal distribution.

6 IMPLEMENTATION

We train the user and interface agent's policies simultaneously in a shared environment (the AUI). All policies receive an independent reward, and the actions of the policies influence a shared environment. We execute actions in the following order: (1) the interface agent's action, (2) the

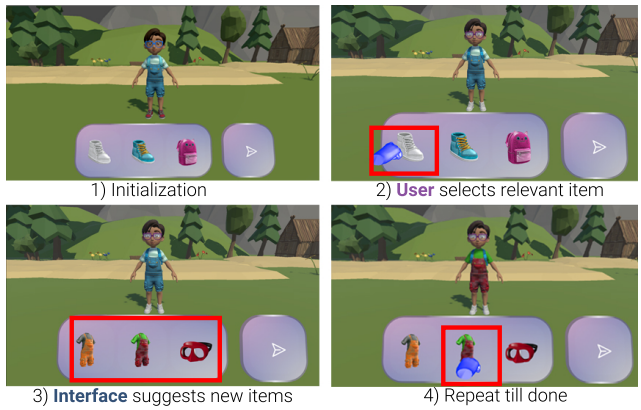

Fig. 3. In our proposed task the interface agent and the user agent interact in a turn-based manner, in which the user (agent) matches a game character selection to a target state (1). First, the user operates a toolbar with three slots (2). Second, The interface agent assigns the most relevant items to the available slots (3). Finally, This cycle continues till the two characters match (4).

user agent's high-level action, followed by (3) the user agent's low-level motor action. The reward for the two learned policies is computed after the low-level motor action has been executed. The episode is terminated when the user agent has either completed the task or exceeded a time limit.

We implement our framework in Python 3.8 using RLLIB [65] and Gym [7]. We use PPO [95, 112] to train our policies. We use 3 cores on an Intel(R) Xeon(R) CPU @ 2.60GHz during training and an NVIDIA TITAN Xp GPU. Training takes ~36 hours. The user agent's high-level decision-making policy π_d is a 3-layer MLP with 512 neurons per layer and ReLU activation functions. The interface agent's policy π_I is a two-layer network with 256 neurons per layer and ReLU activation functions. We sample the full state initialization (including goal) from a uniform distribution. We use stochastic sampling for our exploration-exploitation trade-off.

We use curriculum learning to increase the task difficulty and improve learnability. Specifically, we adjust the difficulty level every time a criteria has been met by increasing the mean number of initial attribute differences. More initial attribute differences result in longer action sequences and are therefore more complex to learn. We increase the mean by 0.01 every time the successful completion rate is above 90% and the last level up was at least 10 epochs away.

We randomly sample the number of attribute differences from a normal distribution with standard deviation 1, normalize the sampled number into the range $[1, n_a]$ and round it to the nearest integer, where n_a is the number of attributes of a setting (in the case of game character $n_a = 5$).

7 EVALUATION

MARLUI aims to learn UI adaptations from simulated users that can support real users in the same task. Specifically, we want our framework to produce AUIs that are competitive with baselines that require carefully collected real user data. In this section, we evaluate if our approach achieves this goal. Thus, we first conduct an *in silico study* to analyze how the interface agent and the user agent solve the UI adaptation problem in simulation. Then, we conduct a *user study* to investigate if policies of the interface agent that were learned in simulation benefit real users in terms of number of actions needed to complete a task.

7.1 Task & Environment

To conduct the evaluation, we introduce the *character-creation task* (see Fig. 3). In this task, a user creates a virtual reality game character by changing its attributes. A character has five distinct attributes with three items per attribute: i) shoes (red, blue, white), ii) shirt (orange, red, blue), iii) glasses (reading, goggles, diving), iv) backpack (pink, blue, red), and v) dance (hip hop, break, silly). The characters' attribute states are limited to one per attribute, i.e., the character cannot be dancing hip hop and break simultaneously. This leads to a total of 15 attribute items and 243 character configurations. We sample the goal uniformly from the different configurations.

The game character's attributes can be changed by selecting the corresponding items in a toolbar-like menu with three slots. The user can cycle through the items by selecting "Next." The static version of the interface has all items of an attribute assigned to the three slots, and every attribute has its own page (e.g., all shoes, if the user presses next, all backpacks). The character's attribute states correspond to the current input state $\mathbf{x}$ and the target state $\mathbf{g}$, where $\mathbf{g}$ is only known to the user agent. The goal of the interface agent is to reduce the number of clicks necessary to change an attribute, by assigning the relevant items to the available menu slots. For the *user agent*, the higher level selects a target slot, and the lower level moves to the corresponding location.

7.2 In Silico

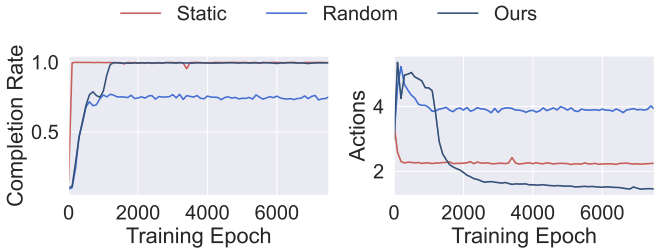

Fig. 4. We train our agent till convergence. Left: the fraction of successfully completed episodes per epoch. Ours and Static reach a 100% successful completion rate. Random does not converge. Right: The number of actions needed on average during a successful episode. Our framework needs less actions compared to Static and Random.

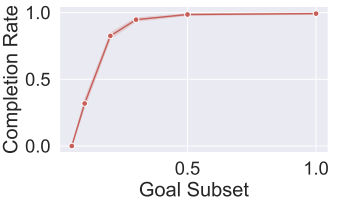

Fig. 5. The fraction of successfully completed episodes as function of the fraction of the total number of goals. The graph shows that it is sufficient to see half of the goals to learn policies that generalize to all goals.

Training. We evaluate the training of our framework against a static and a random interface. In the random interface, items are randomly assigned to the slots. Our method would perform on par, or worse, with random if it was incapable of learning. Hence, the random baseline provides a lower bound for performance. The static baseline has no adaptations, allowing us to evaluate the general effectiveness of our adaptive interface. For most tasks, a task-specific heuristic could be found that presents an upper baseline. However, this has to be designed specifically for each task. On the other hand, our goal is to provide a general framework that works across tasks with minimal changes to the reward function and observation spaces.

Figure 4 shows the user agent's task completion rate and number of actions per task of all three interfaces during training. Ours and the static baseline converge, whereas the random baseline

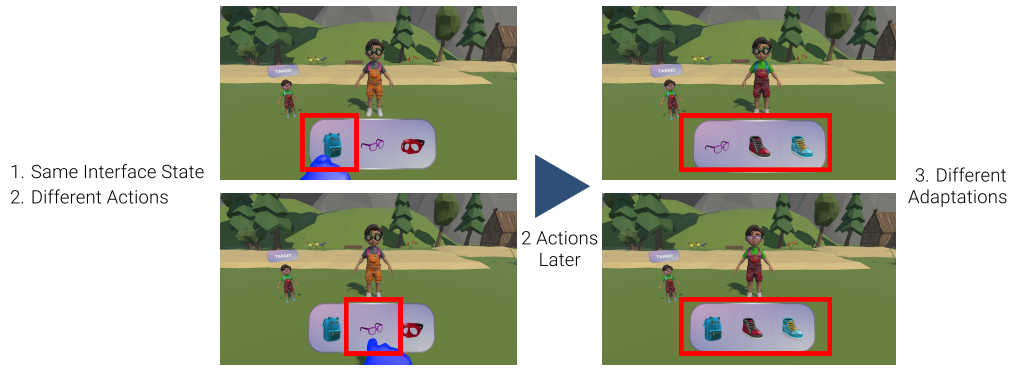

Fig. 6. With our framework, multiple relevant items can be assigned simultaneously; yet the user can only select one (left). Depending on the user's action (top: select backpack, bottom: select glasses), other item gets assigned later (top: shoes, bottom: backpack). Note that the user has not selected any backpack in those two actions. This shows that our framework actively adapts to user input.

does not. Furthermore, the mean number of actions of ours is lower than the mean of the static interface.

Generalization to unseen goals. To understand how well our approach can generalize to unseen goals, we ablate the fraction of goals the agents have access to during training. We then evaluate the learned policies against the full set of goals, which is defined as all possible combinations of character attributes. The results are presented in Figure 5. We find that having access to roughly half of the goals is sufficient to not impact the results. This indicates that our approach generalizes to unseen goals of the same set for this task.

Understanding policy behavior. We qualitatively analyzed the learned policies of our interface agent to understand how it supports the user agent in its task. In Figure 6, we show a snapshot of two sequences with identical initialization. To reach the target character configuration, the user agent can either select the blue bag or the purple glasses (both are needed). Depending on which item the user selects at this time step, the interface agent proposes different suggestions in subsequent steps. For instance, it might happen that the interface agent suggest two relevant items; while the user can only select a single on. As example, the blue backpack the user did not select initially (Figure 6, bottom) gets suggested again later; as the user has not selected a single backpack during the interaction cycle. This behavior shows that the interface agent implicitly reasons about the attribute that the user intends to select based on previous interactions. In short, the interface agent learns to suggest items that the user is not wearing, or that the user has not interacted with.

7.3 User Study

Our goal was to create a user agent whose behavior resembles that of real users, so the interface agent can support them in the same task. To this end, we evaluated the sim-to-real transfer capabilities of our framework by conducting a user study where the interface agent interacted with participants instead of the user agent.

7.3.1 Baselines. We compared our framework to two supervised learning approaches and the static interface (see Sec. 7.1). In line with previous work [34], we used a Support Vector Machine (SVM) with a Radial basis function (RBF) kernel as a baseline. It represents a direct competitor to

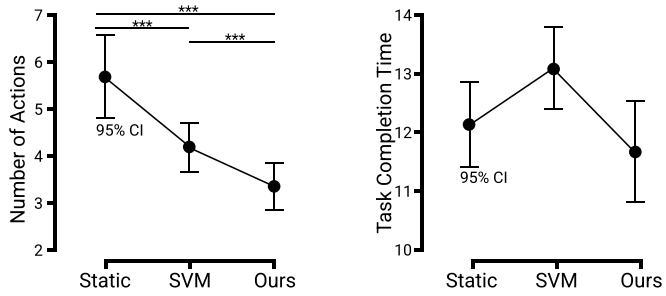

Fig. 7. The average number of actions (left) and the task completion time (right) participants needed to finish the tasks of our user study, plotted with the 95% confidence interval. We compare Ours against Static, and an SVM. Our approach outperforms both baselines on the number of actions. Averaged over all participants and all trials.

our approach, as it generalizes on a method level. However, it needs data recollection for every interface and task.

We used the implementations of scikit-learn [86] and optimized the hyperparameters for performance. The feature vector of the baseline was identical to that of our framework. The baseline learned the probability with which a user will select a certain character attribute next. We assigned the three attributes with the highest probability to the menu slots. Note that we did not consider "Next" to be an item.

Dataset. We collected data from 6 participants to train the supervised baselines. These participants did not take part in the user study. They interacted with the static interface, which resulted in a dataset with over 3000 logged interactions. We saw that the SVM performance saturates when increasing to more than 2700 data points and reached around 91% top-3 classification accuracy (i.e., the percentage of how often the users' selected item was in the top three of the SVM output) on a test set. Furthermore, we found that the baseline generalize well to unseen participants (again through cross-validation). We used all data points for the final model.

Metrics. We used two metrics (dependent variables) to evaluate our approach.

(1) *Number of Actions:* the number of clicks a user needed to complete a task, which is a direct measure of user efficiency [10].
(2) *Task Completion Time:* the total time a user needed to complete a task.

7.3.2 Procedure. Participants interacted with the interface agent and the two baselines. The three settings were counterbalanced with a Latin square design, and the participants completed 30 trials per setting. In each condition, we discarded the first six trials for training. The participants were instructed to solve the task as fast as possible while reducing the number of redundant actions. They were allowed to rest in-between trials. We ensured that the number of initial attribute differences (IAD), which refers to the number of clothing attributes that differ between the current selection and the target at the start of a trial, was uniformly distributed within the participant's trials. Participants used an Oculus Quest 2 with its controller.

We recruited 12 participants from staff and students of an institution of higher education (10 male, 2 female, aged between 23 and 33). All participants were right-handed and had a normal or correct-to-normal vision. On average, they needed between 35 to 40 minutes to complete the study.

Table 1. User study result reported by IAD and method (Mean ± SD).

	1	2	3	4	5
Heuristic	1.708 ± 0.396	3.889 ± 0.451	6.139 ± 0.797	7.903 ± 0.845	9.000 ± 0.000
SVM	1.456 ± 0.219	3.747 ± 0.515	4.607 ± 0.174	5.439 ± 0.259	5.827 ± 0.389
Our	1.625 ± 0.569	**2.036 ± 0.297**	**3.281 ± 0.657**	**4.719 ± 0.613**	5.311 ± 0.186

7.3.3 Results. We present a summary of our results in Figure 7. We analyzed the effect of conditions on the performance of participants with respect to the number of actions and task completion time.

Participants needed on average 3.34 actions to complete a task with our framework, compared to 5.73, and 3.87 for the static, and SVM baselines respectively. The normality and sphericity were not violated. We performed a repeated-measures ANOVA We found a significant effect on method ($F(2, 22) = 209.68, p < .001$). With a holm-corrected post-hoc we find that both the SVM and Ours significantly outperform the static interface (both $p < .001$) in terms of number of actions. We also find that ours significantly outperforms the SVM ($p = 0.006$).

It is important to decompose the results per initial attribute difference. This is crucial because a task with a single IAD is expected to be shorter in duration than tasks with multiple IADs. We report the values in Table 1. Using a repeated-measures ANOVA, we found a significant effect ($F(8, 0.25) = 35.537, p < .001$) on the interaction between the approach and the IAD. We limited our analysis to scenarios where the interaction involves "Ours" and only considered scenarios where the IADs between the methods were identical (i.e., we do not discuss, for example, the SVM with five initial attribute differences versus the static method with a single difference). Our findings from a Holm-corrected post-hoc analysis showed no difference between the static baseline and the other approach for a single initial attribute difference (all $p = 1.00$) and a significant difference for more than one IAD (all $p < .001$). When comparing the SVM versus our method, we found no difference for one IAD (p=1.00) or for five (p=0.26); for all other cases, a difference was found (all $p < .001$). We observed that the interface agent's strategy, which groups similar category items (e.g., showcasing pink and blue backpacks if the character has a red one cf. Fig. 6), increases the chance of selecting a useful item. However, this strategy doesn't apply when all attributes need changing.

When looking at the task completion time, participants using our framework needed 12.14 seconds to complete the task. The completion time was 12.36 for the static interface and 12.71 for the SVM. The task completion time was normally distributed (Shapiro-Wilk $p > 0.05$). We found no significant difference in overall task completion time with a Greenhouse-Geisser (for sphericity) corrected repeated-measures ANOVA ($F(1.568, 17.243) = 0.86, p = 0.42$). This could be, because despite requiring more action for a static baseline, the participants formed a strategy.

7.4 Discussion

To analyze if our multi-agent framework is competitive with a baseline that requires carefully collected real user data, we compared it against a supervised SVM. We did not find significant differences between the two approaches in the task performance metrics of the number of actions and task completion time. This suggests that our approach is a competitive alternative to data-driven approaches for creating adaptive user interfaces.

The adaptive approaches significantly reduce the number of actions necessary to complete the task compared to the static interface. However, no significant differences in task completion times were found. We argue that this could be due to real users being more familiar with the ordering of

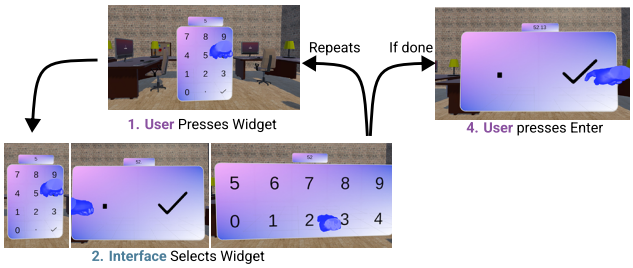

Fig. 8. Adaptive keypad: the user agent is asked to enter a randomly initialized price by using a keypad (1). The interface agent selects from three pre-designed different widgets either a normal keypad, a digits-only keypad or a non-digits-only keypad. The user agent selects a button of the chosen widget. The task ends when the user agent presses enter (4).

items in the static interface that is kept constant across trials. This familiarity is not captured by our current cognitive model or incentivized in the reward function. In future work, we will model familiarity and investigate its effect on task completion time.

We have shown qualitatively that our interface agent learns to take previous user actions into account. This characteristic is core to meaningful adaptations. At the moment, our agent's capabilities are limited by the size of the stack $\mathbf{o}$. In the future, recurrent approaches such as LSTM could be investigated to overcome this limitation.

Furthermore, we presented evidence that our framework can generalize to goals that were not seen during training. It is important to mention that the results of this experiment are subject to its task and that seen and unseen goals are from the same distribution. Nevertheless, the study provides first indications that our approach generalizes to applications where not all users' goals might not always be encountered during training.

8 APPLICATION TO DIFFERENT INTERFACES AND TASKS

To demonstrate the versatility of our framework, we introduce four additional point-and-click interfaces to demonstrate how our approach generalizes to different scenarios. Every scenario offers a distinct adaption, task, and interface. Our method requires minimal to no adaptations for the different scenarios (i.e., mostly a change in the dimensions of the observation and action spaces). We showcase both 2D interfaces as well as Mixed Reality interfaces. We show how the interface agent selects from a set of pre-designed UI widgets, hand user's the correct blocks when building a tower, move out-of-reach items closer, and make hierarchical menus more efficient during photo editing.

Please refer to our supplementary video for visual demonstrations of the tasks. Due to the diverse nature of use cases, we will report either number of clicks or task completion time as success metric. All scenarios are showcased through the interaction of a real user with a trained interface agent. The numerical results are obtain in simulation.

8.1 Number Entry

We introduce a price entry task on a keypad. The interface agent selects a widget from a pre-designed set. We show that our approach can support applications requiring users to issue command sequences and provide meaningful help given a user's progress in the task (see Fig. 8). The task assumes a setting where the simulated user must enter a product price between 10.00 and 99.99. To complete the task, the user agent has to enter the first two digits, the decimal point, the second two

digits, and then press enter. The interface agent can select one of three different interface layouts: i) a standard keypad, ii) a keypad with only digits and iii) a widget with only the decimal point and the enter key.

The goal difference penalty (Eq. 3) in this case is based on whether the current price $\mathbf{x}$ matches the target price $\mathbf{g}$:

$$\mathcal{E}_{gd} = -\sum_{t} \mathbb{1}_{\mathbf{x}_t \neq \mathbf{g}_t}, \tag{8}$$

where $\mathbb{1}$ is an indicator that is 1 if $\mathbf{x}_t \neq \mathbf{g}_t$ and 0 otherwise, and t is the current timestep. Every time a button is hit, t increases by 1. This is similar to the penalty in all other tasks. However, it considers that the order of the entries matters. On average, the user agent needs 4.0 seconds to complete the task in cooperation with the interface agent, compared to 4.9 seconds when using a static keypad. The number of clicks is identical, since the full task can be solved on the standard keypad.

Qualitative Policy Inspection. We observe that the interface agent learns to select the UI that has the biggest buttons for an expected number entry (e.g., only digits or only non-digits). From this we can conclude that the interface agent implicitly learns the concept of Fitt's law and prioritizes larger buttons where appropriate.

8.2 Block Building

The second scenario is a block-building task (Fig. 9) where the user constructs various castle-like structures from blocks. Compared to the game character task, only the dimensionality of the observation and action space needs to be changed. It can choose between 4 blocks (wall, gate, tower, roof) and a delete button. The agent needs to move the hand to a staging place for the blocks (see Figure 9) and then place the block in the corresponding location. The block cannot be placed in the air, i.e., it always needs another block on the floor below. The interface agent suggests a next block every time the user places a block. However, the user can put the block down, in case it is unsuitable. An action is picking or placing a block.

This task represents a subset of tasks that do not have a Heads-Up-Display-like UI to interact with, but are situated directly in the virtual world. This is a common interactive experience of AR/VR systems. The user needs on average 1.1 actions with our framework, compared to 2.0 actions without the interface agent. Thus 1.1 indicates that the interface agent suggests the correct next block, most of the time.

Qualitative Policy Inspection. We observe that the policy learns to always suggest a block that is usable given the current state of the tower. This indicates that the policy has an implicit understanding of the order of blocks and can distinguish between those belonging to the foundation versus the upper parts of a tower.

8.3 Out-of-reach Item Grabbing

In the third usage scenario, the user needs to use their hand to grab an object that is initially out of reach. Thus, the interface agent needs to move an object within reach of the user, which can then grab it. The interface agent observes the location of the user's hand. The task environment includes several objects. Uniquely, in this scenario the user and the interface agent are forced to collaborate to select the correct target object and complete the task (see Figure 10); as it is impossible for the user agent to complete the task on its own. Compared to the game character task, only the dimensionality of the observation and action space needs to be changed.

In this use case, we changed the lower level of our user to learn motor control with RL instead of using the Fitts-Law-based motor controller. This highlights the modularity of our approach and can

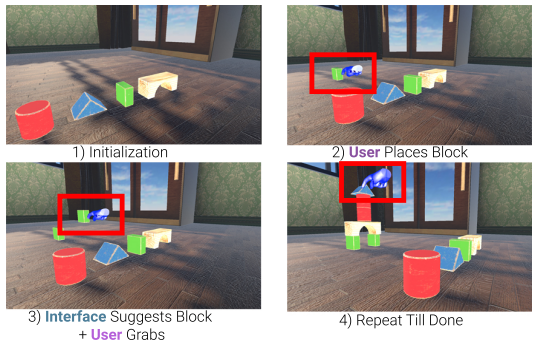

Fig. 9. Block Building: The user is building a castle from blocks (1). The user places the first block (2). The interface agent suggests a next block to place (3). This is repeated till the castle is built (4).

Fig. 10. Out-of-reach object grabbing: the user agent attempts to grab a specific object, that is initially out of reach, in a space containing multiple objects (1). The user agent learned to move towards an object to indicate its intention to grab it (2). Based on that, the interface agent learned to move the intended object within the user agent's reach (3). The user agent then grabs the object to finish the task (4).

be useful in scenarios where existing models, such as Fitts' Law, are not sufficient. The low-level motor control policy controls the hand movement. In particular, given a target slot, the policy selects the parameters of an endpoint distribution. Given the current position and the endpoint parameters (mean and standard deviation), we compute the predicted movement time using the WHo Model [41]. The low-level policy needs to learn i) the coordinates and dimensions of menu slots, ii) an optimal speed-accuracy trade-off given a target slot, and its current position. Refer to Appendix A for more details.

Qualitative Policy Inspection. We find that the the policy selects objects positioned in the direction of the user's arm movement rather than the closest ones. This indicates that the policy implicitly learns the correlation between directionality of movement and intent.

8.4 2D Hierarchical Menu

In this task, a user edits a photo by changing its attributes. A photo has five distinct attributes with three states per attribute: i) filter (color, sepia, gray), ii) text (none, Lorem, Ipsum), iii) sticker (none, unicorn, cactus), iv) size (small, medium large), and v) orientation (original, flipped horizontal, and vertical). The photo's attribute states are limited to one per attribute, i.e., the photo cannot

Fig. 11. We introduce a photo editing task where (1) a user matches a photo to a target by operating a hierarchical menu. (2) The user selects the submenu 'size'. (3) The user then selects the attribute 'small', which alters the image. (4) After the user has changed an attribute, the interface observes the new state of the photo and finds the most likely submenu for the next user action. (5) The user clicks on an item in the submenu to complete the task.

be in grayscale and color simultaneously. This leads to a total of 15 attribute states and 243 photo configurations.

The graphical interface is a hierarchical menu, where each attribute is a top-level menu entry, and each attribute state is in the corresponding submenu. By clicking a top-level menu, the submenu expands and thus becomes visible and selectable. Only one menu can be expanded at any given time.

The photo attribute states correspond to the current input state and the target state. The interface agent selects an attribute menu to open. Its goal is to reduce the number of clicks necessary to change an attribute, e.g., from two user interactions (filter->color) to one (color). For the *user agent*, the higher level selects a target slot, and the lower level moves to the corresponding location.

Qualitative Policy Inspection. We observe that the interface agent intelligently decides which submenu to open next. We notice that this is never a menu the user recently interacted with as the probability of having to change, e.g., the color twice in a row is minimal and only a result of errors.

9 LIMITATIONS & FUTURE WORK

MARLUI models the interaction with point-and-click adaptive interfaces as a multi-agent cooperative game by teaching a simulated user agent and an interface agent to cooperate. Learned policies of the interface agent have shown their capability to effectively assist real users. Demonstrating our approach in a wide variety of use cases is a first step towards general methods that are not tied to specific applications nor dependent on manually crafted rules or offline user data collection. However, there are limitations that require further research.

In this work, we have focused on point-and-click interfaces. However, it would be interesting to extend the user agent to model other interaction paradigms. By enhancing our user agent to replicate behavior for other interaction types than clicking, we could extend the possible use cases that MARLUI can support. For example, research has shown that gaze-based selection, similar to cursor movement, follows Fitts' Law [94]. Furthermore, similar concepts can also be applied to human-robot interactions, such as using simulated humans to train human-robot handshakes or human-to-robot handovers [16, 18, 19].

Moreover, user goals can change during human-computer interaction, particularly in creative tasks where users constantly adjust their objective based on intermediate results. This presents challenges for standard RL approaches, which assume goals to remain stationary. Future research on MARL for AUIs needs to focus on finding strategies to easily adapt trained interfaces to changing or new user goals. This is required to establish more robust and flexible adaptive interfaces that can support real-world use cases.

We have demonstrated that our formulation solves problems with up to 5 billion possible states, as in the character creation application (Sec. 7.1). However, the complexity of the problem grows exponentially with the number of states. This makes it challenging for MARLUI to scale to interfaces with even larger state spaces. To overcome this, we could explore different input modalities, such as representing the state of the UI as an image instead of using one-hot encoding. This approach is similar to work on RL agents playing video games [77], which showed that image representations can effectively cope with large state spaces.

We have shown that the simulated user agent's behavior was sufficiently human-like to enable the interface agent to learn helpful policies that transfer to real users. The interface agent's performance is inherently limited by the user agent. Therefore, increasing realism in the model of the simulated user is an interesting future research direction, for instance, modeling human-like search [14] or motor control with a biomechanical model [27]. Along similar lines, our work helps with the creation of policies that adapt interfaces given user interactions. However, it does not adapt to the user themselves (e.g., different levels of expertise). Such personalization is an interesting direction for future research.

Our framework has theoretical appeal because it provides a plausible model of the bilateral nature of AUIs: the adaptation depends on the user, whereas also the user action depends on the adaptation. Modeling this unilaterally as in supervised learning does not reflect reality well. Related to ongoing research [78], we believe that future work can leverage our framework to gain a better theoretical understanding of how users interact with a UI. Our setup has the potential to scale to multiple users with different skills and intentions. This could lead to bespoke assistive UIs for users with specific needs or UIs for users with specific expertise levels.

Finally, our proposed framework enables adaptive policies for different point-and-click tasks and interfaces. We have shown how our framework produces policies that support users in a variety of these interfaces and tasks. Building on this, future work can investigate the transition from framework to developer tool. Tools that enable developers to use our framework easily and consistently will streamline the development process of AUIs.

10 CONCLUSION

The question addressed in this paper is whether it is possible to develop a *general framework* for point-and-click AUIs that does not depend on task-specific heuristics or data to generate policies offline. To this end, we have introduced MARLUI, a multi-agent reinforcement learning approach. Our method features a user agent and an interface agent. The user agent aims to achieve a task-dependent goal as quickly as possible, while the interface agent learns the underlying task structure by observing the interactions between the user agent and the UI. Since the user agent is RL-based and thus learns through trial-and-error interactions with the interface, it does not require real user data. We have evaluated our approach in simulation and with participants, by replacing the user agent with real users, across five different interfaces and various underlying task structures. The tasks ranged from assigning items to a toolbar, handing out-of-reach objects to the user, selecting the best-performing interface, providing the correct object to the user, and enabling more efficient interaction with a hierarchical menu. Results show that our framework enables the development of AUIs with minimal adjustments while being able to assist real users in their task. We believe that MARLUI, and a multi-agent perspective in general, is a promising step towards tools for developing adaptive interfaces, thereby reducing the overhead of developing adaptive strategies on an interface- and task-specific basis.

REFERENCES

[1] John R Anderson, Michael Matessa, and Christian Lebiere. 1997. ACT-R: A theory of higher level cognition and its relation to visual attention. *Human–Computer Interaction* 12, 4 (1997), 439–462.

[2] Karl Johan Åström. 1965. Optimal control of Markov processes with incomplete state information. *Journal of mathematical analysis and applications* 10, 1 (1965), 174–205.

[3] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528* (2019).

[4] Pauline M Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. 2011. PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 4 (2011), 1–22.

[5] Wauter Bosma and Elisabeth André. 2004. Exploiting Emotions to Disambiguate Dialogue Acts. In *Proceedings of the 9th International Conference on Intelligent User Interfaces* (Funchal, Madeira, Portugal) *(IUI '04)*. Association for Computing Machinery, New York, NY, USA, 85–92. https://doi.org/10.1145/964442.964459

[6] Matthew Michael Botvinick. 2012. Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology* 22, 6 (2012), 956–962.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:1606.01540 [cs.LG]

[8] Dermot Browne, Peter Totterdell, and Mike Norman. 2016. *Adaptive user interfaces*. Elsevier.

[9] S Card, T Moran, and A Newell. 1983. T he Psychology of Human Computer Interaction.

[10] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1980. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Commun. ACM* 23, 7 (jul 1980), 396–410. https://doi.org/10.1145/358886.358895

[11] Stuart. K. Card, Thomas. P. Moran, and Allen Newell. 1986. The model human processor- An engineering model of human performance. *Handbook of perception and human performance.* 2, 45–1 (1986).

[12] Noshaba Cheema, Laura A Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[13] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. ACM, 456–464. https://doi.org/10.1145/3289600.3290999

[14] Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. 2015. The emergence of interactive behavior: A model of rational menu search. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 4217–4226.

[15] Yi Fei Cheng, Christoph Gebhardt, and Christian Holz. 2023. InteractionAdapt: Interaction-driven Workspace Adaptation for Situated Virtual Reality Environments. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[16] Sammy Christen, Lan Feng, Wei Yang, Yu-Wei Chao, Otmar Hilliges, and Jie Song. 2023. SynH2R: Synthesizing Hand-Object Motions for Learning Human-to-Robot Handovers. *arXiv preprint arXiv:2311.05599* (2023).

[17] Sammy Christen, Lukas Jendele, Emre Aksan, and Otmar Hilliges. 2021. Learning Functionally Decomposed Hierarchies for Continuous Control Tasks With Path Planning. *IEEE Robotics and Automation Letters* 6, 2 (2021), 3623–3630. https://doi.org/10.1109/LRA.2021.3060403

[18] Sammy Christen, Stefan Stevšić, and Otmar Hilliges. 2019. Guided deep reinforcement learning of control policies for dexterous human-robot interaction. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2161–2167.

[19] Sammy Christen, Wei Yang, Claudia Pérez-D'Arpino, Otmar Hilliges, Dieter Fox, and Yu-Wei Chao. 2023. Learning human-to-robot handovers from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9654–9664.

[20] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A Predictive Model of Menu Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 627–636. https://doi.org/10.1145/1240624.1240723

[21] Peng Dai, Christopher Lin, Mausam Mausam, and Daniel Weld. 2013. POMDP-based control of workflows for crowdsourcing. *Artificial Intelligence* 202 (09 2013), 52–85. https://doi.org/10.1016/j.artint.2013.06.002

[22] Quentin Debard, Jilles Steeve Dibangoye, Stéphane Canu, and Christian Wolf. 2020. Learning 3D Navigation Protocols on Touch Interfaces with Cooperative Multi-agent Reinforcement Learning. In *Machine Learning and Knowledge Discovery in Databases*, Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet (Eds.). Springer International Publishing, Cham, 35–52.

[23] Stephanie Denison, Elizabeth Bonawitz, Alison Gopnik, and Thomas L Griffiths. 2013. Rational variability in children's causal inferences: The sampling hypothesis. *Cognition* 126, 2 (2013), 285–300.

[24] Andrew T Duchowski, Krzysztof Krejtz, Izabela Krejtz, Cezary Biele, Anna Niedzielska, Peter Kiefer, Martin Raubal, and Ioannis Giannopoulos. 2018. The index of pupillary activity: Measuring cognitive load vis-à-vis task difficulty with pupil oscillation. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

[25] Andrew Faulring, Brad Myers, Ken Mohnkern, Bradley Schmerl, Aaron Steinfeld, John Zimmerman, Asim Smailagic, Jeffery Hansen, and Daniel Siewiorek. 2010. Agent-Assisted Task Management That Reduces Email Overload. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (Hong Kong, China) *(IUI '10)*. Association for Computing Machinery, New York, NY, USA, 61–70. https://doi.org/10.1145/1719970.1719980

[26] Stefano Ferretti, Silvia Mirri, Catia Prandi, and Paola Salomoni. 2014. Exploiting reinforcement learning to profile users and personalize web pages. In *2014 IEEE 38th International Computer Software and Applications Conference Workshops*. IEEE, 252–257.

[27] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports* 11, 1 (2021), 1–15.

[28] Paul M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47, 6 (1954), 381.

[29] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems* 29 (2016).

[30] Michael J Frank and David Badre. 2012. Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: computational analysis. *Cerebral cortex* 22, 3 (2012), 509–526.

[31] Milica Gašić and Steve Young. 2014. Gaussian processes for POMDP-based dialogue manager optimization. *IEEE Transactions on Audio, Speech and Language Processing* 22, 1 (2014), 28–40. https://doi.org/10.1109/TASL.2013.2282190

[32] Daniel Gaspar-Figueiredo, Silvia Abrahão, Emilio Insfrán, and Jean Vanderdonckt. 2023. Measuring User Experience of Adaptive User Interfaces using EEG: A Replication Study. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*. 52–61.

[33] Daniel Gaspar-Figueiredo, Marta Fernández-Diego, Silvia Abrahao, and Emilio Insfran. 2023. A Comparative Study on Reward Models for UI Adaptation with Reinforcement Learning. *methods* 13 (2023), 14.

[34] Christoph Gebhardt, Brian Hecox, Bas van Opheusden, Daniel Wigdor, James Hillis, Otmar Hilliges, and Hrvoje Benko. 2019. Learning Cooperative Personalized Policies from Gaze Data. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3332165.3347933

[35] Christoph Gebhardt and Otmar Hilliges. 2021. Optimal Control to Support High-Level User Goals in Human-Computer Interaction. In *Artificial Intelligence for Human Computer Interaction: A Modern Approach*. Springer, 33–72.

[36] Christoph Gebhardt, Antti Oulasvirta, and Otmar Hilliges. 2021. Hierarchical Reinforcement Learning as a Model of Human Task Interleaving. *Computational Brain and Behavior* (2021). https://arxiv.org/pdf/2001.02122.pdf

[37] Daniel Gerber, Urwashi Kapasiya, Lukas Rosenbauer, and Jörg Hähner. 2023. Automation of User Interface Testing by Reinforcement Learning-Based Monkey Agents. In *International Conference on Complex Computational Ecosystems*. Springer, 3–15.

[38] Samuel J Gershman, Eric J Horvitz, and Joshua B Tenenbaum. 2015. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science* 349, 6245 (2015), 273–278.

[39] Samuel J Gershman, Edward Vul, and Joshua B Tenenbaum. 2012. Multistability and perceptual inference. *Neural computation* 24, 1 (2012), 1–24.

[40] Dorota Glowacka. 2019. Bandit algorithms in recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 574–575.

[41] Yves Guiard and Olivier Rioul. 2015. A mathematical description of the speed/accuracy trade-off of aimed movement. In *Proceedings of the 2015 British HCI Conference*. 91–100.

[42] Tanay Gupta and Julien Gori. 2023. Modeling reciprocal adaptation in HCI: a Multi-Agent Reinforcement Learning Approach. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–6.

[43] William E Hick. 1952. On the rate of gain of information. *Quarterly Journal of experimental psychology* 4, 1 (1952), 11–26.

[44] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The LumièRe Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (Madison, Wisconsin) *(UAI'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 256–265.

[45] Ronald A Howard. 1960. Dynamic programming and markov processes. (1960).

[46] Andrew Howes, Xiuli Chen, Aditya Acharya, and Richard L Lewis. 2018. Interaction as an emergent property of a Partially Observable Markov Decision Process. *Computational interaction* (2018), 287–310.

[47] Zehong Hu, Yitao Liang, Jie Zhang, Zhao Li, and Yang Liu. 2018. Inference aided reinforcement learning for incentive mechanism design in crowdsourcing. In *Advances in Neural Information Processing Systems (NIPS '18)*. 5508–5518.

https://arxiv.org/abs/1806.00206

[48] Zool Hilmi Ismail and Nohaidda Sariff. 2018. A Survey and Analysis of Cooperative Multi-Agent Robot Systems: Challenges and Directions. In *Applications of Mobile Robots*, Efren Gorrostieta Hurtado (Ed.). IntechOpen, Rijeka, Chapter 1. https://doi.org/10.5772/intechopen.79337

[49] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364, 6443 (2019), 859–865. https://doi.org/10.1126/science.aau6249

[50] Bonnie E John and Dario D Salvucci. 2005. Multipurpose prototypes for assessing user interfaces in pervasive computing systems. *IEEE pervasive computing* 4, 4 (2005), 27–34.

[51] Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. Touchscreen Typing as Optimal Supervisory Control. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM. https://userinterfaces.aalto.fi/touchscreen-typing/

[52] Jussi PP Jokinen, Tuomo Kujala, and Antti Oulasvirta. 2021. Multitasking in driving as optimal adaptation under uncertainty. *Human factors* 63, 8 (2021), 1324–1341.

[53] Ioannis Kangas, Maud Schwoerer, and Lucas Bernardi. 2022. Scalable User Interface Optimization Using Combinatorial Bandits. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3375–3379.

[54] Charles C Kemp, Cressel D Anderson, Hai Nguyen, Alexander J Trevor, and Zhe Xu. 2008. A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In *Proceedings of the 3rd ACM/IEEE international conference on human robot interaction*. 241–248.

[55] Davis E Kieras and Davis E Meyer. 1997. An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human–Computer Interaction* 12, 4 (1997), 391–438.

[56] Janin Koch, Andrés Lucero, Lena Hegemann, and Antti Oulasvirta. 2019. May AI? Design ideation with cooperative contextual bandits. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[57] Sucheta V Kolekar, Sriram G Sanjeevi, and DS Bormane. 2010. Learning style recognition using artificial neural network for adaptive user interface in e-learning. In *2010 IEEE International conference on computational intelligence and computing research*. IEEE, 1–5.

[58] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-powered parameter analysis for visual design exploration. *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14* (2014), 65–74. https://doi.org/10.1145/2642918.2647386

[59] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2016. SelPh : Progressive Learning and Support of Manual Photo Color Enhancement. *Proc. of CHI '16* (2016). https://doi.org/10.1145/2858036.2858111

[60] Melchor Lafuente, Sonia Elizondo, Unai J Fernández, and Asier Marzo. 2023. Comparing a Mid-air Two-Hand Pinching Point-and-Click Technique with Mouse, Keyboard and TouchFree. In *XXIII International Conference on Human Computer Interaction*. 1–4.

[61] Thomas Langerak, Sammy Christen, Anna Maria Feit, and Otmar Hilliges. 2021. Generalizing User Models through Hybrid Hierarchical Control. (2021).

[62] Yezdi Lashkari, Max Metral, and Pattie Maes. 1997. Collaborative interface agents. *Readings in agents* (1997), 111–116.

[63] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037* (2017).

[64] Katri Leino, Kashyap Todi, Antti Oulasvirta, and Mikko Kurimo. 2019. Computer-Supported Form Design Using Keystroke-Level Modeling with Reinforcement Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 85–86. https://doi.org/10.1145/3308557.3308704

[65] Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. arXiv:1712.09381 [cs.AI]

[66] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. 2015. DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*. 591–599. https://arxiv.org/abs/1401.1880

[67] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-aware online adaptation of mixed reality interfaces. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 147–160.

[68] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep reinforcement learning based recommendation with explicit user-item interactions modeling. *arXiv preprint arXiv:1810.12027* (2018). https://arxiv.org/abs/1810.12027

[69] J Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Ken Koedinger, and Emma Brunskill. 2016. Interface design optimization as a multi-armed bandit problem. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 4142–4153.

[70] Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. 2020. Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*.

[71] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[72] Wendy Mackay. 2000. Responding to cognitive overload: Co-adaptation between users and technology. *Intellectica* 30, 1 (2000), 177–193.

[73] Pattie Maes. 1995. Agents that reduce work and information overload. In *Readings in human–computer interaction*. Elsevier, 811–821.

[74] Eric McCreath, Judy Kay, and Elisabeth Crawford. 2006. IEMS-an approach that combines handcrafted rules with learnt instance based rules. *Aust. J. Intell. Inf. Process. Syst.* 9, 1 (2006), 40–53.

[75] Abhinav Mehrotra and Robert Hendley. 2015. Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. (2015), 813–824.

[76] Nesrine Mezhoudi and Jean Vanderdonckt. 2021. Toward a task-driven intelligent GUI adaptation by mixed-initiative. *International Journal of Human–Computer Interaction* 37, 5 (2021), 445–458.

[77] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[78] Roderick Murray-Smith, Antti Oulasvirta, Andrew Howes, Jörg Müller, Aleksi Ikkala, Miroslav Bachinski, Arthur Fleig, Florian Fischer, and Markus Klar. 2022. What simulation can do for HCI research. *Interactions* 29, 6 (2022), 48–53.

[79] Jun Ota. 2006. Multi-agent robot systems as distributed autonomous systems. *Advanced Engineering Informatics* 20, 1 (2006), 59–70. https://doi.org/10.1016/j.aei.2005.06.002

[80] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, and Andreas Karrenbauer. 2020. Combinatorial Optimization of Graphical User Interface Designs. *Proc. IEEE* 108, 3 (2020), 434–464. https://doi.org/10.1109/JPROC.2020.2969687

[81] Antti Oulasvirta, Samuli De Pascale, Janin Koch, Thomas Langerak, Jussi Jokinen, Kashyap Todi, Markku Laine, Manoj Kristhombuge, Yuxi Zhu, Aliaksei Miniukovich, et al. 2018. Aalto interface metrics (AIM) a service and codebase for computational GUI evaluation. In *Adjunct Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 16–19.

[82] Antti Oulasvirta, Anna Feit, Perttu Lähteenlahti, and Andreas Karrenbauer. 2017. Computational Support for Functionality Selection in Interaction Design. 24, 5, Article 34 (oct 2017), 30 pages. https://doi.org/10.1145/3131608

[83] Antti Oulasvirta, Jussi PP Jokinen, and Andrew Howes. 2022. Computational Rationality as a Theory of Interaction. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[84] Antti Oulasvirta, Per Ola Kristensson, Xiaojun Bi, and Andrew Howes. 2018. *Computational interaction*. Oxford University Press.

[85] Seonwook Park, Christoph Gebhardt, Roman Rädle, Anna Maria Feit, Hana Vrzakova, Niraj Ramesh Dayama, Hui-Shyong Yeo, Clemens N Klokmose, Aaron Quigley, Antti Oulasvirta, et al. 2018. Adam: Adapting multi-user interfaces for collaborative environments in real-time. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–14.

[86] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[87] Veljko Pejovic and Mirco Musolesi. 2014. InterruptMe: Designing Intelligent Prompting Mechanisms for Pervasive Applications. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2014), 897–908. https://doi.org/10.1145/2632048.2632062

[88] Athanasios S Polydoros and Lazaros Nalpantidis. 2017. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems* 86, 2 (2017), 153–173.

[89] Derek Reilly, Michael Welsman-Dinelle, Colin Bate, and Kori Inkpen. 2005. Just point and click? Using handhelds to interact with paper maps. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*. 239–242.

[90] Charles Rich, Candy Sidner, Neal Lesh, Andrew Garland, Shane Booth, and Markus Chimani. 2005. DiamondHelp: A collaborative interface framework for networked home appliances. In *25th IEEE International Conference on Distributed Computing Systems Workshops*. IEEE, 514–519.

[91] Charles Rich and Candace L Sidner. 1998. COLLAGEN: A collaboration manager for software interface agents. In *Computational Models of Mixed-Initiative Interaction*. Springer, 149–184.

[92] Fabio Rizzoglio, Maura Casadio, Dalia De Santis, and Ferdinando A. Mussa-Ivaldi. 2021. Building an adaptive interface via unsupervised tracking of latent manifolds. *Neural Networks* 137 (2021), 174–187. https://doi.org/10.1016/j.neunet.2021.01.009

[93] Dario D Salvucci. 2001. An integrated model of eye movements and visual encoding. *Cognitive Systems Research* 1, 4 (2001), 201–220.

[94] Immo Schuetz, T Scott Murdison, Kevin J MacKenzie, and Marina Zannoli. 2019. An Explanation of Fitts' Law-like Performance in Gaze-Based Selection Tasks Using a Psychophysics Approach. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[95] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[96] Young-Woo Seo and Byoung-Tak Zhang. 2000. A reinforcement learning agent for personalized information filtering. In *Proceedings of the 5th international conference on Intelligent user interfaces*. 248–251.

[97] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

[98] Lloyd S Shapley. 1953. Stochastic games. *Proceedings of the national academy of sciences* 39, 10 (1953), 1095–1100.

[99] Jianqiang Shen, Erin Fitzhenry, and Thomas G Dietterich. 2009. Discovering frequent work procedures from resource connections. In *Proceedings of the 14th international conference on Intelligent user interfaces*. 277–286.

[100] Jianqiang Shen, Jed Irvine, Xinlong Bao, Michael Goodman, Stephen Kolibaba, Anh Tran, Fredric Carl, Brenton Kirschner, Simone Stumpf, and Thomas G Dietterich. 2009. Detecting and correcting user activity switches: algorithms and interfaces. In *Proceedings of the 14th international conference on Intelligent user interfaces*. 117–126.

[101] KGGH Silva, WAPS Abeyasekare, DMHE Dasanayake, TB Nandisena, Dharshana Kasthurirathna, and Archchana Kugathasan. 2021. Dynamic user interface personalization based on deep reinforcement learning. In *2021 3rd International Conference on Advancements in Computing (ICAC)*. IEEE, 25–30.

[102] Dustin A Smith and Henry Lieberman. 2010. The why UI: using goal networks to improve user interfaces. In *Proceedings of the 15th international conference on Intelligent user interfaces*. 377–380.

[103] Harold Soh, Scott Sanner, Madeleine White, and Greg Jamieson. 2017. Deep sequential recommendation for personalized adaptive user interfaces. In *Proceedings of the 22nd international conference on intelligent user interfaces*. 589–593.

[104] Constantine Stephanidis, Charalampos Karagiannidis, and Adamantios Koumpis. 1997. Decision making in intelligent user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*. 195–202.

[105] Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130* (2017). https://arxiv.org/abs/1707.00130

[106] Richard S Sutton, Andrew G Barto, et al. 1998. Introduction to reinforcement learning. (1998).

[107] Zheng Tian, Shihao Zou, Ian Davies, Tim Warr, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. 2020. Learning to communicate implicitly by actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7261–7268.

[108] Kashyap Todi, Gilles Bailly, Luis Leiva, and Antti Oulasvirta. 2021. Adapting User Interfaces with Model-based Reinforcement Learning. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM. https://userinterfaces.aalto.fi/adaptive/

[109] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. 2020. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7293–7300.

[110] Zhibo Yang, Lihan Huang, Yupei Chen, Zijun Wei, Seoyoung Ahn, Gregory Zelinsky, Dimitris Samaras, and Minh Hoai. 2020. Predicting goal-directed human attention using inverse reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 193–202.

[111] Neil Yorke-Smith, Shahin Saadati, Karen L Myers, and David N Morley. 2012. The design of a proactive personal agent for task management. *International Journal on Artificial Intelligence Tools* 21, 01 (2012), 1250004.

[112] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems* 35 (2022), 24611–24624.

[113] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).

[114] Chao Yu, Minjie Zhang, Fenghui Ren, and Guozhen Tan. 2015. Emotional Multiagent Reinforcement Learning in Spatial Social Dilemmas. *IEEE Transactions on Neural Networks and Learning Systems* 26, 12 (2015), 3083–3096. https://doi.org/10.1109/TNNLS.2015.2403394

[115] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.

[116] Lamia Zouhaier, Yousra Ben Daly Hlaoui, and Leila Ben Ayed. 2021. A reinforcement learning based approach of context-driven adaptive user interfaces. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 1463–1468.

A LEARNED LOWER LEVEL

We detail how to learn the low-level motor policy. The low-level policy needs to learn i) the coordinates and dimensions of menu slots, ii) an optimal speed-accuracy trade-off given a target slot, and its current position.

To prevent the low-level motor control policy from correcting wrong high-level decisions and to increase general performance, we limit the state space S_M to strictly necessary elements with respect to the motor control task [17]:

$$S_M = (\mathbf{p}, \mathbf{t}), \tag{9}$$

with the current position $\mathbf{p} \in I^2$, the target slot $\mathbf{t} \in \mathbb{Z}_2^{n_s}$.

The action space A_M is defined as follows:

$$A_M = \left(\mu_\mathbf{p}, \sigma_\mathbf{p}\right). \tag{10}$$

It consists of $\mu_\mathbf{p} \in I^2$ and $\sigma_\mathbf{p} \in I$, i.e., the mean and standard deviation which describes the endpoint distribution in the unit interval. We scale the standard deviation linearly between a min and max value where the minimum value is the size of normalized pixel width and the max value is empirically chosen to be 15% of the screen width. Once an action is taken, we sample a new end-effector position from a normal distribution: $\mathbf{p} \sim \mathcal{N}\left(\mu_\mathbf{p}, \sigma_\mathbf{p}\right)$.

Given the predicted actions, we compute the expected movement time via the WHo model [41], similar to our non-learned low-level motor control policy in the main paper.

The reward for the low-level motor control policy is based on the *motoric* speed-accuracy trade-off. Specifically, we penalize: i) missing the target supplied by the high-level ($\neg h$), and ii) the movement time (T_M). Furthermore, we add a penalty iii) which amounts to the squared Euclidean distance between the center of the target $\mathbf{t}$ and $\mu_\mathbf{p}$. This incentivizes the policy to hit the desired target in the correct location. Since the penalty only considers the desired point $\mu_\mathbf{p}$, it will not impact the speed-accuracy trade-off (which is a function of $\sigma_\mathbf{p}$). The total reward is defined as follows:

$$R_M = \underbrace{\lambda(\neg h)}_{i)} - \underbrace{(1-\lambda)T_M}_{ii)} - \underbrace{\beta||\mu_\mathbf{p} - \mu_\mathbf{t}||_2^2}_{iii)}, \tag{11}$$

where $\neg h$ equals 0 when the target button is hit and -1 on a miss. A hit occurs when the newly sampled user position $\mathbf{p}$ is within the target $\mathbf{t}$, while a miss happens if the user position is outside of the target. λ is a speed-accuracy trade-off weight and β is a small scalar weight to help with learning.