Speeding up Inference with User Simulators through Policy Modulation

Hee-Seung Moon
Yonsei University
Republic of Korea
hs.moon@yonsei.ac.kr

Seungwon Do
ETRI
Republic of Korea
seungwon.do@etri.re.kr

Wonjae Kim
NAVER AI Lab
Republic of Korea
wonjae.kim@navercorp.com

Jiwon Seo*
Yonsei University
Republic of Korea
jiwon.seo@yonsei.ac.kr

Minsuk Chang*
NAVER AI Lab
Republic of Korea
minsuk.chang@navercorp.com

Byungjoo Lee*
Yonsei University
Republic of Korea
byungjoo.lee@yonsei.ac.kr

ABSTRACT

The simulation of user behavior with deep reinforcement learning agents has shown some recent success. However, the inverse problem, that is, *inferring the free parameters of the simulator from observed user behaviors*, remains challenging to solve. This is because the optimization of the new action policy of the simulated agent, which is required whenever the model parameters change, is computationally impractical. In this study, we introduce a network modulation technique that can obtain a generalized policy that immediately adapts to the given model parameters. Further, we demonstrate that the proposed technique improves the efficiency of user simulator-based inference by eliminating the need to obtain an action policy for novel model parameters. We validated our approach using the latest user simulator for point-and-click behavior. Consequently, we succeeded in inferring the user's cognitive parameters and intrinsic reward settings with less than 1/1000 computational power to those of existing methods.

CCS CONCEPTS

• **Human-centered computing** → **User models**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

simulation model, inverse modeling, point-and-click

ACM Reference Format:
Hee-Seung Moon, Seungwon Do, Wonjae Kim, Jiwon Seo, Minsuk Chang, and Byungjoo Lee. 2022. Speeding up Inference with User Simulators through Policy Modulation. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3491102.3502023

*Corresponding authors.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3502023

1 INTRODUCTION

User performance is a foundational factor when designing interfaces. If we can predict user performance over variations of interfaces through mathematical modeling, it will allow designers to rapidly evaluate, or even optimize, the interfaces. Advances in machine learning techniques and the increase in computational power over the past decade have opened new opportunities in research on user performance modeling, allowing researchers to address the high dimensionality, variability, and adaptability of human behavior. In particular, studies on *simulation models* of user performance have made significant progress in recent years. Simulation models attempt to explain the user's behavior as an integrated system of several sub-processes; therefore, they have a much larger number of model parameters than traditional user performance models and can benefit from the recent development of computational techniques. In recent studies, simulation of button pressing [50], point-and-click [19], typing behavior [31], layout learning [32], menu search [15], and mid-air movement [14, 23] are good examples of new possibilities for modern user simulation modeling.

Similar to the other models, the simulation model can be used in two ways. The first is generative use, predicting the variables of an interaction in which we are interested. This is similar to how Fitts' law [24, 69] can provide insight into interface design by predicting the user's target selection time. Second, the simulation model is used to infer user and interface characteristics by fitting the model to the given interaction data. From this *inverse modeling*, the main keyword of this study, we can estimate the *free parameters* of the model that represent the user and interface characteristics [34]. For example, we can estimate the throughput and y-intercept by fitting Fitts' law to the target selection time data. By referring to the obtained parameters, it was possible to optimize graphical user interfaces [49] and evaluate the performance of pointing devices [20].

In general, simulation models have a significant number of free parameters that have more explicit physical meaning because, unlike traditional descriptive models, the mechanism behind user behavior must be precisely reproduced as a combination of sub-processes. For example, a simulation model of point-and-click behavior recently published by Do *et al.* [19] includes 15 free parameters representing the characteristics of the user and the system, such as signal-dependent motor noise [68], visual perception noise [72], precision of internal clock [40, 77], click success reward, and click failure penalty. Therefore, if we succeed in the inverse modeling of a

simulation model, we can obtain other rich insights about users and interfaces that are not possible with traditional performance models. However, inverse modeling using a modern simulation model has not been widely attempted because of a significant technical bottleneck, as described below.

The basic principle of inverse modeling is to determine the free parameters of the model that best describe the given data. Consequently, it is necessary to search for a high-dimensional free parameter space and find the parameter set that maximizes the likelihood of the given data. The problem here is that, generally, when the free parameters change, the simulated agent's *action policy* must also be updated [34]. Action policy is a decision-making function that determines the action the agent will take in a specific task situation. In modern user simulation models, the user's action policy is expressed by a neural network that directly maps a task state to an action or one that predicts the utility of all available actions (the latter is the so-called Q-network; however, for simplicity, both are referred to as a policy network in this paper). The connection weights of the policy network can be determined through computational techniques such as reinforcement learning on the premise that the user has an optimal policy [26, 42], which takes several hours to a few days with a normal computer [19]. Consequently, an iterative search in the free parameter space becomes impractical.

In this study, we present a novel technique that can overcome the bottleneck in the inverse modeling of user simulators. The core of the technique is to obtain a generalized action policy of the simulated user that can immediately reflect changes in the free parameters in simulations. Accordingly, we design a policy model using a neural network architecture that can be *conditioned* (i.e., *modulated*) on the given free parameters of the simulator. More specifically, our policy model adapts its mapping function by modulating the intermediate feature values (i.e., hidden states) of the network by feature-level concatenation or FiLM [54]. Consequently, a user simulator equipped with our generalized policy model can exhibit optimal behavior for any given free parameter (e.g., different cognitive parameters or reward formulation). Therefore, the cost of the iterative search for inverse modeling can be drastically reduced because the trained user simulator can now adapt its behavioral strategy with no further policy optimization.

We demonstrated our proposed technique using a *point-and-click task* as an example, in which a state-of-the-art simulation model was recently published [19]. Point-and-click involves the selection of a distant target (stationary or moving) by controlling the cursor with an indirect pointing device such as a computer mouse. To perform a point-and-click task, users must visually perceive the state of the cursor and target (visual perception), plan and execute cursor movement (motor control), and decide when to perform a click (click decision-making). It is also known that this process is influenced by the speed–accuracy bias instruction given to the user. For example, a user's point-and-click varies significantly between asking them to click as fast as possible and to click as accurate as possible [83]. In relation to these processes, we inferred the following six free parameters of the state-of-the-art point-and-click simulation model [19]: precision of visual speed perception (σ_v), coefficient of signal-dependent motor noise (n_v), precision of click decision-making (c_σ), reward weights for successful click ($w_{success}$), motor effort (w_{effort}), and elapsed time (w_{time}).

We conducted the inference process in two parts: (1) three reward weights (i.e., $w_{success}$, w_{effort}, and w_{time}) were inferred at the population level, and then, (2) three cognitive parameters (i.e., σ_v, n_v, and c_σ) were inferred at the individual level. We implemented simulation models equipped with our modulated action policy for each part, enabling the inference with significantly reduced computational cost. For each part, we evaluated the generalization performance of our modulated action policy by verifying whether our policy model can sufficiently approximate the simulation results from multiple individually trained policy models. To evaluate the inference performance, we collected point-and-click behavioral data from 20 participants and measured the baseline values of their cognitive parameters in controlled experiments. For the first part of the inference, we examined whether changes in the inferred reward weights seem plausible according to the changes in the speed–accuracy bias instruction given to the user. Consequently, we could reasonably estimate the intrinsic reward settings of users. For the second part, we examined the correlation between the inferred and measured cognitive parameters of each participant. Consequently, we showed that two of the three targeted cognitive parameters (σ_v and c_σ) could be estimated with a moderate level of coefficient of determination (R^2=0.50 for σ_v; R^2=0.61 for c_σ). This inference at the individual level with many users has been infeasible in previous studies. With our method of improving the efficiency, the user simulator-based inference, which previously required hundreds or thousands of hours, is now accomplished in a few hours.

To the best of our knowledge, no study has inferred multiple free parameters of reinforcement learning-based user simulators as efficiently as ours. We expect that our proposed technique can be widely used in interface personalization, optimization research, and recommendation system research. We released all these datasets as open sources for future research[1]. The contributions of this study can be summarized as follows:

- We proposed a generalized policy model implementation method that can significantly improve the efficiency of the inverse modeling of a user simulator.
- We collected point-and-click behavioral data for multiple users (N=20) and measured the baseline values of their cognitive characteristics (i.e., visual perception noise, motor noise, and precision of click decision-making). The dataset is released as an open-source and can serve as a benchmark dataset for user simulator-based inverse modeling studies.
- We demonstrated an inference process based on our proposed method. We succeeded to infer multiple cognitive parameters and intrinsic reward settings of users from their point-and-click behaviors, even with significantly reduced computational costs.

2 RELATED WORK

2.1 Simulation Model of User Behavior

Traditionally, user performance models have focused on the prediction of aggregated performance variables that summarize interactions over a period, such as trial completion time [2, 69], error rate [79], accuracy [40], and precision [27]. Similar to the well-known

[1] https://github.com/hsmoon121/pnc-dataset

Fitts' law [24], such models exhibit high robustness in predicting the user performance on target tasks; however, they cannot explain the cognitive processes through which the user performs the task over time. Conversely, simulation models of user behavior predict not only the aggregated performance variables of traditional models but also how the state of the user will change over time [19]. Previous simulation models of user behavior first appeared in the 1980s. GOMS [12] models predicted user behavior over time by considering the individual execution time of each cognitive process. Cognitive architectures, such as ACT-R [4] and EPIC [36], enabled more sophisticated user simulation by modeling cognitive mechanisms, such as memory retrieval or parallel operations of perceptual-motor modules.

Previous user simulation approaches are limited in that they require precise descriptions of the user's sub-behavior to accomplish the sub-goals of the task (e.g., production rules), which were mostly hand-coded by modelers. As an alternative to the hand-coded rules, users' sequential decision-making behavior can be modeled by learning an *action policy*, which is a decision-making function that determines the user's action from the current task state. Such approaches build on the consensus that users behave to maximize their expected utility [8, 26, 42, 52]. That is, a user's behavior can be assumed as optimal behavior within the possible behavioral space bounded by human capabilities (e.g., cognitive characteristics). Reinforcement learning (RL) can be applied to achieve the optimal behavior of an agent in an interactive environment, which is formulated as a Markov decision process (MDP). In the MDP setting, an agent can take *action* in the current *state* and observe a *reward* and the next *state*, and through RL, the agent's decision-making strategy (i.e., an action policy) is optimized. Previous attempts have been made to apply traditional RL methods (e.g., Q-learning) for user simulations, such as simulating eye movement [70] or dialog management [41]. However, the traditional RL methods are not suitable for solving problems with high-dimensional state and action spaces; therefore, the previous approaches were also limited to addressing simple MDP problems.

In the era of deep learning, the use of neural networks and reinforcement learning techniques has brought significant improvements in finding the optimal behavior of agents in a wide range of tasks, from playing video games [45] to acquiring physics-based motion skills [53]. In deep RL, an action policy is computed using neural networks and becomes suitable to manage environments with high-dimensional spaces. Therefore, simulation models have recently begun applying deep RL to achieve user behavior in performing complex human–computer interaction (HCI) tasks. Recent approaches simulated user behavior as an integrated system of sub-modules that reflect human capabilities and an optimized action policy that governs the operation of the modules. For example, Cheema *et al.* [14] reproduced user behavior of performing a mid-air pointing task by biomechanically modeling the human upper limb and optimizing an action policy, which determines the joint torque, to minimize fatigue of the upper limb and pointing error. Other recent studies, such as user simulation of button pressing [50], touch screen typing [31], menu search [15], visual search [33], layout learning [32], reaching movement [23], and point-and-click behavior [19], also exhibited a wide range of HCI situations in which the RL-based approach can be used.

While the application field of the modern simulation model broadens, there is an open question as to how to effectively address the wide variability of behavior across individual users. Individual-level user models are expected to achieve better prediction performance than a model fitted to the entire user pool [34, 46, 47]. However, in previous approaches, the implementation of individual-level simulation models consumed time and computing resources and thus was often impractical. This is because, if the free parameters of the user simulator change (i.e., the behavioral space bounded by human capabilities changes), the action policy of the simulated user must be optimized. In this study, we aimed to solve this bottleneck to enable individual-level simulation. Our key approach is to implement a generalized action policy that can reflect the variations in the simulated user's behavioral or cognitive characteristics (i.e., the free parameters of the user simulator).

2.2 Policy Modulation Techniques

Changes in the free parameters of the user simulator can be considered as changes in the MDP formulation that the simulated agent faces. Therefore, our attempt to generalize the action policy is to solve a family of MDPs using a single policy network, which can be interpreted as solving the following two types of RL problems. First, if the cognitive characteristics of a simulated user change, the probabilistic transition function between states of the MDP changes accordingly. Therefore, training an action policy that can respond to variations in cognitive characteristics can be regarded as *multi-task* RL [37, 81, 82], which aims to train an agent's policy to operate in multiple task environments. Second, we consider the intrinsic reward formulation of a simulated user, which governs their behavior in completing HCI tasks. Responding to the variations in the simulated user's reward formulation can be regarded as *multi-objective* RL [1, 66, 80], which aims to generalize an agent's policy across several objectives in a task environment, thereby exhibiting optimal behavior for any given objective condition.

To address these RL problems, a policy network should be *modulated* according to *context* parameters that contain information of a given task or objective, thereby changing its mapping function depending on the given context. Methods to modulate (or *condition*) neural networks can be classified into the following three levels: The first method involves simply including context parameters in the input of the network (i.e., conditioning on the *input*). Rakelly *et al.* [57] generalized the policy network over multiple tasks by inputting the latent representation of the task identity (i.e., task embedding) to the policy. Second, context parameters can be used to condition the intermediate *features* (i.e., hidden state) of the networks. One method is to concatenate context parameters to the hidden states (i.e., feature-level concatenation), as in recent studies that adapt a single policy network to multiple objectives [1] or tasks [87]. The hidden states can also be modulated by linear transformation (i.e., scaling and shifting) by relying on feature-wise linear modulation (FiLM) [54]. In FiLM, a separate network (i.e., FiLM generator) is trained with a primary network (e.g., policy network), and context parameters are mapped to coefficients that scale and shift the hidden states through the trained FiLM generator. Recent RL studies have demonstrated that the use of FiLM can allow the agent policy to be adjusted by task instructions [6] or adapted to

multiple task environments [76]. Finally, the *weights* of the networks can be entirely conditioned by given the context parameters. A hypernetwork [29] is a structure that enables such weight-level conditioning; that is, a secondary network (hypernetwork) takes a conditioning input (context parameters) and produces the weights of a primary network. In general, a hypernetwork entails more parameter changes in a primary network compared to the previous two conditioning levels; therefore, a higher modulation capacity at the cost of higher computational complexity is expected. Owing to its complexity, research on hypernetwork structure in the RL field is still underway; however, there are a few recent studies related to the continual learning of task dynamics [30] or multi-agent RL [58].

Although approaches to implement RL-based simulation models have become diverse, such policy modulation techniques have not yet been introduced in the HCI field. In this study, we propose a method to implement a policy network of a simulation model that can be modulated by varying the free parameters. Among the three levels of modulations, we consider feature-level modulation techniques (feature-level concatenation or FiLM) to provide (1) enhanced modulation capacity compared to the input-level modulation and (2) more stability during optimization compared to weight-level modulation. With the modulation techniques, our implemented simulation model can immediately adapt its behavior to any given free parameter.

2.3 Inverse Modeling for HCI Research

Inverse modeling of user simulation or performance models (i.e., inversely estimating the free parameters of the model from the given interaction data) can provide rich insight into the interaction design because the free parameters represent the characteristics of the user and the interaction environment. Typically, inverse modeling can be performed by loss minimization [21]. For example, we can devise a function that calculates the discrepancy (e.g., root mean squared deviation or χ^2) between the model's prediction and the given data, and to determine the parameters of the model that minimize the discrepancy (least-squares estimation). If the model includes more advanced probabilistic processes, we can determine the parameters of the model that maximize the likelihood of observing the given dataset (maximum likelihood estimation, MLE). In addition to these, if the likelihood function of a probabilistic model is difficult to compute, simulation-based inference techniques such as approximate Bayesian computation (ABC) [35] or Bayesian optimization for likelihood-free inference (BOLFI) [28] can be applied.

Because traditional user performance models generally have a small number of parameters and assume a simple stochastic process, successful inverse modeling was possible based on least-squares estimation. For example, the free parameters of Fitts' law or Steering law have been estimated for different input devices [3, 5, 10, 56], different body parts [38, 71], different operational biases [83, 85], and users of different age groups [9, 84]. More complicated stochastic models of user behavior include the drift-diffusion model for the reaction process [59] and Stocker's model for speed perception [72]. Regarding the drift-diffusion model, there is a study comparing the strengths and weaknesses of various inverse modeling techniques including MLE [60]. In the case of Stocker's model, parameters were estimated based on MLE in the original paper [72]. Eye movements and movement of attention (EMMA) model [61] describes the movement of gaze and visual attention, and the model parameters in the original paper were manually tuned to mimic human data. This hand-tuned MLE is also frequently observed in more advanced simulation models such as ACT-R [25, 62–64], which clearly shows that computational techniques for inverse modeling of user simulators have not been widely attempted until the early 2000s [34, 78].

Recently published RL-based simulation models on user behavior [14, 15, 19, 32] have approximately 10 free parameters on average, and in most studies, the values were imported directly from previous studies or hand-tuned. In 2017, Kangasrääsiö *et al.* [34] first attempted inverse modeling of an RL-based user simulator [15] through BOLFI. Kangasrääsiö *et al.* estimated the posterior distribution of the free parameters (e.g., users' duration of fixations) given the observation dataset of the users' menu search behavior. Through an iterative search in the parameter space, BOLFI discovered the free parameters that exhibit the least discrepancy between the simulation and observation. Consequently, the estimated parameters improved the model fit compared to using manually tuned parameters in the original model [15]. The study deserves attention in that it first introduced a principled method to infer the parameters of RL-based simulation models in HCI. However, the time inefficiency problem of the iterative search remained; a single inference process took hundreds of hours (in CPU time) in [34], because it involved the process of newly optimizing an action policy for each new free parameter sample. The required computational time may increase to thousands of hours for simulation models dealing with more complex HCI tasks, thus making the inference impractical. With our proposed method, an action policy can be adapted according to any given free parameter without further optimization. Therefore, our method can facilitate the inverse modeling of user simulators by dramatically reducing the time required for the iterative search of free parameters.

3 INFERENCE WITH A GENERALIZED USER BEHAVIOR SIMULATOR

Given a reinforcement learning-based user behavior simulation model, we propose a technique to optimally maintain the simulated user behavior even if the free parameters of the simulated users are changed (policy modulation). A more efficient inverse modeling process using such a generalized simulation model was also demonstrated. In this section, we introduce a general formulation of RL-based user simulators and the implementation of our proposed technique is described in detail.

3.1 RL-based User Behavior Simulator

In general, an RL-based user simulator is implemented as follows: First, the simulated user's cognitive and behavioral processes required to complete the target task and the characteristics of the given environment are mathematically modeled. This model usually includes free parameters representing the characteristics of the simulated user and the environment (e.g., cognitive capabilities, reward weights for different objectives, or setting of input devices). Then, the intrinsic decision-making process of the simulated user, which is involved in the cognitive and behavioral processes, can be formulated as MDP. Under the MDP formulation, the *action policy*

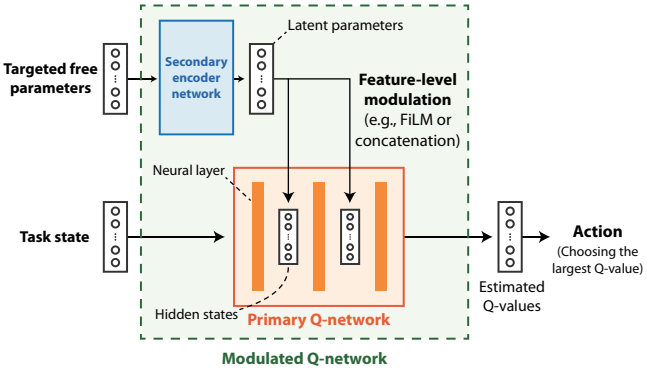

Figure 1: The structure of the modulated Q-network.

of the simulated user determines the proper next action according to the current observation of the task environment (i.e., task state) at every decision-making step.

In particular, in recent user simulation model studies [15, 19, 31, 33], the optimal action policy of the simulated user is achieved by estimating the *Q-values*. The Q-value, an RL terminology, represents the expected cumulative reward until the episode ends when a corresponding action is taken at the current state. The action policy can be implemented as a neural network model, the so-called *Q-network*, which predicts the Q-values of all available actions from the given task state. The Q-network can be optimized through deep RL algorithms, such as deep Q-network (DQN) [45], and by relying on the trained Q-network, a simulated user can retain the optimal policy of selecting the action with the largest predicted Q-value.

Previous studies have obtained the optimal policy (i.e., trained a Q-network) when the free parameters of a simulated user are fixed to specific values (i.e., values of an average person). Therefore, whenever the characteristics of the simulated user or task environment are changed, the policy must be newly optimized.

3.2 Policy Modulation

3.2.1 Modulated Q-network. We introduce a method to generalize the optimal policy of the simulation model to the variations in the free parameters, that is, enabling policy modulation. Among the known modulation approaches (Section 2.2), we empirically found in this study that the best modulation performance for the latest point-and-click simulation model [19] is achieved when the targeted free parameters are involved in the intermediate features of the Q-network (i.e., feature-level modulation). The choice of the modulation method is crucial to achieve optimal policy modulation of simulation models, and our empirical finding for the point-and-click simulation model may not be directly applicable to other simulation models. In this study, we present a *modulated Q-network*, which is a network structure that enables feature-level modulation by providing the targeted free parameters as auxiliary inputs.

The presented structure consists of two neural networks: a primary Q-network and a secondary encoder network (Figure 1). The primary network predicts the Q-values of all available actions from a given task state. The primary Q-network can change the mapping to Q-values through feature-level modulation according to the latent parameters. The secondary network (i.e., encoder network) generates the latent parameters; the secondary network receives the targeted free parameters and outputs their latent representation. Through the training process (RL) along with the primary Q-network, the secondary network learns to extract the information from the free parameters required for the effective modulation of the primary Q-network.

We consider two representative methods of feature-level modulation that have been proven suitable for modulation in recent RL studies: feature-level concatenation [1, 87] and FiLM [6, 54, 76]. Concatenating the latent parameters into hidden states prevents dilution of the information of latent parameters as it passes through neural layers; therefore, the primary Q-network can effectively incorporate the conditioning information to predict the Q-values. FiLM provides a more direct method to modulate hidden states through linear transformation (scaling and shifting). In the case of using FiLM, the latent parameters are employed as shifting and scaling coefficients applied to each hidden state; that is, the secondary network acts as the FiLM generator in [54]. The modulation method (e.g., feature-level concatenation or FiLM) and the structure of the networks (e.g., width and depth of hidden layers) can be changed depending on the types of targeted free parameters because the abstraction process required to achieve optimal modulation may differ according to each type of free parameter.

3.2.2 Training method. To train the modulated Q-network, we extended the previous RL algorithms that train a Q-network (the DQN family). In this section, we describe the training method with an example of the original DQN [45]; however, this can be equally applied to the DQN family. For example, in Sections 7.1.2 and 8.1.2, we apply two different DQN-based algorithms, namely, the envelope multi-objective Q-learning (MOQL) [80] and double DQN [74]. During the training phase of DQN, decision processes of the agent at every timestep (a tuple of MDP transition, (s_t, a_t, r_t, s_{t+1}), representing the state, action, reward at timestep t, and state at timestep t+1, respectively) are stored in the *replay* memory. At every training step, the Q-network of the agent is updated by employing the batch of the transitions sampled from the replay memory, in the direction of minimizing the temporal difference (TD) error defined as follows:

$$r + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t),$$

where γ denotes the discount factor of the MDP, $Q(s, a)$ denotes the estimated Q-value by the Q-network. As training proceeds, the Q-network accurately estimates the Q-values and the agent acquires the optimal behavior.

For the generalization of the action policy, the targeted free parameters (denoted as z) are newly sampled at the beginning of each training episode. That is, in each episode, the simulated user explores a task environment with different free parameters (e.g., different cognitive characteristics or reward formulations). The sampled free parameters are stored in the replay memory along with each transition of the episode; that is, the experience tuple is extended as $(s_t, a_t, r_t, s_{t+1}, z)$. A batch of the extended experiences is used to calculate the TD error of the estimation from the modulated Q-network as follows:

$$r + \gamma \max_{a'} Q(s_{t+1}, a'|z) - Q(s_t, a_t|z),$$

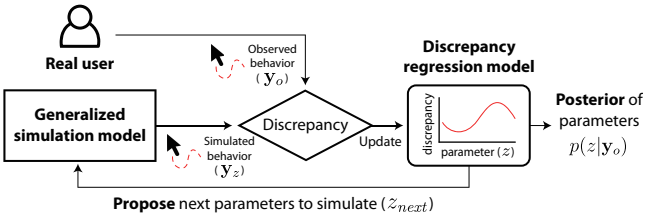

Figure 2: Overview of the BOLFI process to infer free parameters of the simulation model (z) corresponding to a real user. BOLFI constructs a regression model that estimates the discrepancy between the user's observed behavior ($\mathbf{y}_o$) and the simulated behavior of the model ($\mathbf{y}_z$) with a given z. BOLFI proceeds with the following iterative process: (1) BOLFI proposes the next parameter sample, z_{next}, that is likely to lead to low discrepancy, based on the regression model; (2) the regression model is updated with the discrepancy value obtained from the simulation based on z_{next}. In this paper, we implemented a generalized simulation model over variations in z. Therefore, there is no need to optimize the simulation model for every proposed z. After sufficient iterations, BOLFI estimates the posterior of z for the given observed behavior, $p(z|\mathbf{y}_o)$.

where $Q(s, a|z)$ denotes the estimated Q-value by the modulated Q-network, which is conditioned on the sampled free parameters z. Through backpropagation, the modulated Q-network (i.e., the primary Q-network and secondary encoder network) can be optimized. A framework such as the prioritized experience replay (PER) [67], which can increase the training efficiency based on the TD error of each experience tuple, can also be applied to the training with no further modification.

3.3 Inferring Free Parameters with User Simulator

The simulated behavior reflecting any given free parameters now can be obtained without the need to re-train the action policy; that is, the simulation model is generalized over the free parameters. Next, with the generalized simulation model, we are facing the inverse modeling problem, that is, inferring the free parameters from a real user's observed behavior.

Simulation-based inference methods (e.g., ABC [35]) provide a principled method to infer the free parameters by systematically searching the parameters that best describe the observed behavior of the simulation. In this study, we applied BOLFI [28],a recent simulation-based inference method (also a variant of ABC), which was first introduced in HCI in [34]. In the latest attempt on inverse modeling [34], hundreds of hours (in CPU time) were required for a single inference because every single simulation for given simulation parameters entails the process of newly training the action policy. With no re-training process, our generalized simulation model can significantly reduce the computational cost of the inference process.

The BOLFI procedure is shown in Figure 2. BOLFI requires: a simulation model $\mathcal{M}$, which reproduces the behavioral data $\mathbf{y}_z$ given simulation parameters z; observed behavioral data $\mathbf{y}_o$; and a

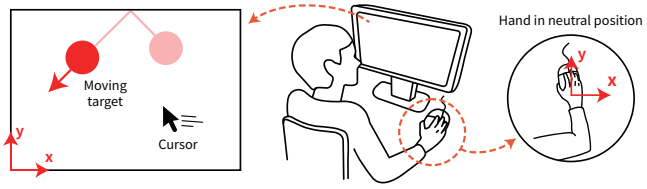

Figure 3: The point-and-click task environment covered in this study. *Center*: A right-handed user performs a point-and-click task on screen with a mouse device. *Left*: The xy coordinate of a target and a cursor on the screen. *Right*: The xy coordinate of a user's hand.

function to mesasure the discrepancy between $\mathbf{y}_z$ and $\mathbf{y}_o$, denoted as $\Delta(z)$. The inference goal is to determine z showing the least $\Delta(z)$. The essence of BOLFI is to estimate the discrepancy ($\Delta(z)$) according to the given simulation parameters by constructing a regression model (via Gaussian process). Whenever $\mathbf{y}_z$ is simulated using a new z, BOLFI updates the regression model based on the evaluated discrepancy. Our generalized simulation model can immediately obtain $\mathbf{y}_z$ for the new z with no further policy optimization; therefore, we can significantly improve the time efficiency compared to previous approaches. The learned regression model is used to propose the next z to simulate (z_{next}). At every iteration, BOLFI chooses the z_{next} following its acquisition rule—usually, z with the minimum lower confidence bound value of the predicted discrepancy. This allows the simulation to be conducted mainly in the low-discrepancy region, therefore reducing the number of simulations required for the inference. After running sufficient simulations, BOLFI estimates a posterior of z that shows the least $\Delta(z)$ given $\mathbf{y}_o$, and we can use maximum a posteriori (MAP) estimation to obtain the exact values of the inferred free parameters. More details of BOLFI can be found in the original paper [28].

4 POINT-AND-CLICK SIMULATION MODEL

We demonstrate the performance of the proposed technique using the latest user behavior simulator implemented based on deep RL [19]. In particular, the simulation model realistically simulated the *point-and-click* behavior of users. The model has 15 free parameters regarding the user's physical characteristics, cognitive characteristics, and intrinsic reward settings. In this section, the model is briefly introduced.

4.1 Point-and-Click Scenario

The simulation model assumes a specific point-and-click scenario. The user uses a computer mouse to control the cursor on the screen. A circular target is moving at a constant velocity on the screen; when the target hits the edge of the screen, it changes the direction of movement as if reflected, and the speed is maintained (Figure 3). The user is right-handed and the distance between the user's head and the display is 63 cm. The user is supposed to press the mouse button when the cursor is within the target. When the user presses the mouse button, the trial ends regardless of whether the target is successfully acquired, and a new target appears with a random velocity at a random location.

different intrinsic reward formulation. Furthermore, a participant's reward formulation may change as they progress through point-and-click trials. If the user-independent and time-invariant reward formulation is assumed as in this study, the actual difference in reward formulation may not be reflected in the simulated behavior.

Second, we assumed that the cognitive parameters measured through the cognitive experiments in Study 1 were transferred to point-and-click tasks while maintaining their values. However, in reality, the values of the cognitive parameters may change as the target task changes. There can also be variations depending on the user. This may lead to discrepancies between the inferred and measured baseline values of the cognitive parameters.

Third, the point-and-click model [19] may not perfectly reproduce the *cursor trajectory* in reality although the model is known to faithfully reproduce the user's *task performance*. In this study, we used the discrepancy between the the observed and simulated cursor trajectories for inference. More considerations than the current simulation model may be needed to accurately describe user behavior at the trajectory level. For example, in consecutive trials, the user can use a strategy of preparing the next trial whenever each trial ends (e.g., moving the cursor to the neutral position); however, the current simulation model does not consider this preparation process. In addition, participants often exhibited suboptimal behaviors that could not be explained by the simulation model (e.g., modifying their mouse grips during a trial). Such discrepancies between the simulation model and the user's behavioral characteristics lead to a difficult situation for precise inference.

Furthermore, we assumed that the point-and-click model had the same reproducibility over a defined range of free parameter values. In the original paper [19], the authors verified the model's reproducibility in the case when setting the free parameters according to the typical values reported in previous studies; however, it has not been validated that the reproducibility remains the same over a wide range of parameter values. As the problem of needing a certain upper bound of σ_v was discovered (Section 6.2.2), the precision of the simulation may decrease as the determined free parameters become far from the typical values, and this may lead to another potential bias that degrades inference performance.

Finally, as a potential reason for the unsatisfactory inference of motor noise (n_v), n_v may not cause a significant difference in the point-and-click behaviors between users; therefore, it is difficult to infer n_v based on the observations of user behaviors. This explanation can be supported by the observation from real users in Study 1, in that we could not find a significant correlation between the measured n_v and the participant's task performance. In addition, as shown in Figure 16, we confirmed that the change in the motor noise did not significantly influence the action policy of the simulation models. This result matches with the previous study [19], which conducted an ablation study of the point-and-click model and showed that the effect of motor noise on the simulated behavior was less significant than that of visual perception noise.

9 CONCLUSION AND FUTURE WORK

In this study, we proposed and validated an implementation method for a generalized simulation model that can significantly reduce the computational cost of the inverse modeling of user simulators. Contrary to the previous approaches that required iterative RL processes for every new free parameter, our method enabled the simulated user's policy to immediately adapt to given free parameters without additional optimization; therefore, the efficiency of inverse modeling can be significantly improved (e.g., hundreds or thousands of hours to only a few hours per single inference).

We verified the proposed method by applying it to the latest point-and-click simulation model. We inferred the intrinsic reward settings of participants from their point-and-click behaviors; that is, we can now plausibly explain the changes in their behavioral strategies under different levels of speed–accuracy trade-off. We also inferred each participant's cognitive parameters (visual perception noise and click precision) involved in their cognitive processes related to point-and-click tasks. To our best knowledge, this study enables practical inference of the cognitive parameters of individual users based on a point-and-click simulation model for the first time.

There are several promising directions for future research. First, we independently trained the two types of generalized simulation models for the inference of reward weights and cognitive parameters in this study, by assuming that all participants had the same reward formulation. Because of this simplification, the inference performance may be degraded. However, if the simulation model on both reward weights and cognitive parameters can be generalized, it becomes possible to simultaneously infer an individual user's intrinsic reward settings and cognitive parameters from the observed user behaviors without simplification. This simultaneous inference of the reward weights and cognitive parameters is a challenging problem because as the number of free parameters that a model needs to generalize increases, the required training costs (e.g., time and computational resources) and modulation capacity of a network increase as well. Thus, a network structure with a higher modulation capacity (e.g., hypernetworks [29]) can be considered; however, it may lead to unstable learning.

Second, although our framework significantly enhanced the computational efficiency of the inverse modeling of user simulators, there was an inevitable time consumption for the iterative search of the simulation parameter space. Accordingly, the enabling of *real-time* inference of a user's cognitive parameters is still an open research question. One possible approach is *amortized* inference [18, 55], which constructs a surrogate model that receives behavioral data and estimates the posterior of cognitive parameters. If the training of the surrogate model can be performed only with simulation models, real-time inference based on the observed user behaviors can become possible using the trained surrogate model.

Finally, the applicability of our efficient inverse modeling approach to simulation models in other HCI tasks can be investigated. One research question that can arise when applying our approach to other HCI tasks is whether our modulated Q-network is also effective to implement simulation models in other MDP problems. The feature-level modulation technique has been validated in previous RL studies [1, 6, 76, 87]; however, it is still unclear how the presented structure works in higher-dimensional action and state space than that of point-and-click tasks; therefore, it would be valuable to verify its generalization ability. When dealing with such complex MDP problems, more computation is required to abstract the task state or free parameters; therefore, a deeper structure of the Q-network might be necessary.

If our method can enable practical inference of a user's intrinsic rewards or cognitive characteristics in a wide range of HCI tasks, it is expected to obtain richer insights from the user's behaviors and utilize them for various applications. For example, the inferred cognitive parameters of a user can be used as a basis for interface optimization or personalization. If the interface is adapted to the user's cognitive characteristics (e.g., ability-based optimization [65]), the usability of the interface for each user can be improved. In addition, assuming that the user's cognitive characteristics are retained and transferred to a related task [73], we can predict the same user's performance on similar tasks. For example, in the gaming field, the prediction of a player's performance based on inferred cognitive characteristics can provide valuable insight for difficulty design; it becomes possible to provide the most suitable difficulty level to the player, such that the player can be fully immersed [17]. An efficient inverse modeling approach can also be extended to quickly grasp a user's personal preferences or intentions. Here, a user's inferred preferences or predicted next action can be used to improve recommender systems.

ACKNOWLEDGMENTS

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07043580), in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2020R1A2C400214612), and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (2020-0-01361).

REFERENCES

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*. 11–20.

[2] Johnny Accot and Shumin Zhai. 1997. Beyond Fitts' law: Models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. 295–302.

[3] Johnny Accot and Shumin Zhai. 1999. Performance evaluation of input devices in trajectory-based tasks: An application of the steering law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 466–472.

[4] John R Anderson. 1996. ACT: A simple theory of complex cognition. *American Psychologist* 51, 4 (1996), 355.

[5] Daniel Bachmann, Frank Weichert, and Gerhard Rinkenauer. 2015. Evaluation of the leap motion controller as a new contact-free pointing device. *Sensors* 15, 1 (2015), 214–233.

[6] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. 2018. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations*.

[7] Gilles Bailly, Antti Oulasvirta, Duncan P Brumby, and Andrew Howes. 2014. Model of visual search and selection time in linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3865–3874.

[8] Sheldon Baron and David L Kleinman. 1969. The human as an optimal controller and information processor. *IEEE Transactions on Man-Machine Systems* 10, 1 (1969), 9–17.

[9] George Erich Brogmus. 1991. Effects of age and sex on speed and accuracy of hand movements: And the refinements they suggest for Fitts' law. *Proceedings of the Human Factors Society Annual Meeting* 35, 3 (1991), 208–212.

[10] Rachael A Burno, Bing Wu, Rina Doherty, Hannah Colett, and Rania Elnaggar. 2015. Applying Fitts' law to gesture based computer interactions. *Procedia Manufacturing* 3 (2015), 4342–4349.

[11] Robin T Bye and Peter D Neilson. 2008. The BUMP model of response planning: Variable horizon predictive control accounts for the speed–accuracy tradeoffs and velocity profiles of aimed movement. *Human Movement Science* 27, 5 (2008), 771–798.

[12] Stuart K Card, Thomas P Moran, and Allen Newell. 1983. *The psychology of human-computer interaction*. Lawrence Erlbaum Associates.

[13] Géry Casiez and Nicolas Roussel. 2011. No more bricolage! Methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. 603–614.

[14] 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.

[15] 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.

[16] Ian D Colley, Peter E Keller, and Andrea R Halpern. 2018. Working memory and auditory imagery predict sensorimotor synchronisation with expressively timed music. *Quarterly Journal of Experimental Psychology* 71, 8 (2018), 1781–1796.

[17] Anna Cox, Paul Cairns, Pari Shah, and Michael Carroll. 2012. Not doing but thinking: The role of challenge in the gaming experience. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 79–88.

[18] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. 2020. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences* 117, 48 (2020), 30055–30062.

[19] Seungwon Do, Minsuk Chang, and Byungjoo Lee. 2021. A simulation model of intermittently controlled point-and-click behaviour. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–17.

[20] Sarah A Douglas, Arthur E Kirkpatrick, and I Scott MacKenzie. 1999. Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 215–222.

[21] Simon Farrell and Stephan Lewandowsky. 2018. *Computational modeling of cognition and behavior*. Cambridge University Press.

[22] Jocelyn Faubert. 2002. Visual perception and aging. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale* 56, 3 (2002), 164.

[23] 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.

[24] 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.

[25] Wai-Tat Fu and Peter Pirolli. 2007. SNIF-ACT: A cognitive model of user navigation on the World Wide Web. *Human–Computer Interaction* 22, 4 (2007), 355–412.

[26] 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.

[27] 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.

[28] Michael U Gutmann, Jukka Corander, et al. 2016. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research* (2016).

[29] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. In *International Conference on Learning Representations*.

[30] Yizhou Huang, Kevin Xie, Homanga Bharadhwaj, and Florian Shkurti. 2020. Continual model-based reinforcement learning with hypernetworks. *arXiv preprint arXiv:2009.11997* (2020).

[31] 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*. 1–14.

[32] Jussi PP Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling learning of new keyboard layouts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4203–4215.

[33] Jussi PP Jokinen, Zhenxin Wang, Sayan Sarcar, Antti Oulasvirta, and Xiangshi Ren. 2020. Adaptive feature guidance: Modelling visual search with graphical layouts. *International Journal of Human-Computer Studies* 136 (2020), 102376.

[34] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. 2017. Inferring cognitive models from data using approximate Bayesian computation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1295–1306.

[35] Antti Kangasrääsiö, Jussi PP Jokinen, Antti Oulasvirta, Andrew Howes, and Samuel Kaski. 2019. Parameter inference for computational cognitive models with Approximate Bayesian Computation. *Cognitive Science* 43, 6 (2019), e12738.

[36] 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.

[37] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526.

[38] Gary D Langolf, Don B Chaffin, and James A Foulke. 1976. An investigation of Fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior* 8, 2 (1976), 113–128.

[39] Byungjoo Lee, Sunjun Kim, Antti Oulasvirta, Jong-In Lee, and Eunji Park. 2018. Moving target selection: A cue integration model. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[40] Byungjoo Lee and Antti Oulasvirta. 2016. Modelling error rates in temporal pointing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1857–1868.

[41] Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1998. Using Markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*. 201–204.

[42] Richard L Lewis, Andrew Howes, and Satinder Singh. 2014. Computational rationality: Linking mechanism and behavior through bounded utility maximization. *Topics in Cognitive Science* 6, 2 (2014), 279–311.

[43] Ray F Lin and Yi-Chien Tsai. 2015. The use of ballistic movement as an additional method to assess performance of computer mice. *International Journal of Industrial Ergonomics* 45 (2015), 71–81.

[44] I Scott MacKenzie and Poika Isokoski. 2008. Fitts' throughput and the speed-accuracy tradeoff. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1633–1636.

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[46] Hee-Seung Moon and Jiwon Seo. 2021. Fast User Adaptation for Human Motion Prediction in Physical Human–Robot Interaction. *IEEE Robotics and Automation Letters* 7, 1 (2021), 120–127.

[47] Hee-Seung Moon and Jiwon Seo. 2021. Optimal Action-based or User Prediction-based Haptic Guidance: Can You Do Even Better?. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[48] Andrew Y Ng and Stuart J Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning*. 663–670.

[49] 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.

[50] Antti Oulasvirta, Sunjun Kim, and Byungjoo Lee. 2018. Neuromechanics of a button press. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[51] Eunji Park and Byungjoo Lee. 2020. An Intermittent Click Planning Model. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[52] Stephen J Payne and Andrew Howes. 2013. Adaptive interaction: A utility maximization approach to understanding human interaction with technology. *Synthesis Lectures on Human-Centered Informatics* 6, 1 (2013), 1–111.

[53] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. 2016. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics* 35, 4 (2016), 1–12.

[54] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[55] Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. 2020. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[56] Robert G Radwin, Gregg C Vanderheiden, and Mei-Li Lin. 1990. A method for evaluating head-controlled computer input devices using Fitts' law. *Human Factors* 32, 4 (1990), 423–438.

[57] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*. 5331–5340.

[58] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*. 4295–4304.

[59] Roger Ratcliff. 1978. A theory of memory retrieval. *Psychological Review* 85, 2 (1978), 59.

[60] Roger Ratcliff and Francis Tuerlinckx. 2002. Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction times and parameter variability. *Psychonomic Bulletin & Review* 9, 3 (2002), 438–481.

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

[62] Dario D Salvucci. 2001. Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies* 55, 1 (2001), 85–107.

[63] Dario D Salvucci. 2006. Modeling driver behavior in a cognitive architecture. *Human Factors* 48, 2 (2006), 362–380.

[64] Dario D Salvucci, Yelena Kushleyeva, and Frank J Lee. 2004. Toward an ACT-R general executive for human multitasking. In *Proceedings of the Sixth International Conference on Cognitive Modeling*. 267–272.

[65] Sayan Sarcar, Jussi PP Jokinen, Antti Oulasvirta, Zhenxin Wang, Chaklam Silpasuwanchai, and Xiangshi Ren. 2018. Ability-based optimization of touchscreen interactions. *IEEE Pervasive Computing* 17, 1 (2018), 15–26.

[66] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *International Conference on Machine Learning*. 1312–1320.

[67] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *International Conference on Learning Representations*.

[68] Richard A Schmidt, Howard Zelaznik, Brian Hawkins, James S Frank, and John T Quinn Jr. 1979. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review* 86, 5 (1979), 415.

[69] R William Soukoreff and I Scott MacKenzie. 2004. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *International Journal of Human-Computer Studies* 61, 6 (2004), 751–789.

[70] Nathan Sprague and Dana Ballard. 2003. Eye movements for reward maximization. In *Advances in Neural Information Processing Systems*. 1467–1474.

[71] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. 2015. Investigating the dexterity of multi-finger input for mid-air text entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3643–3652.

[72] Alan A Stocker and Eero P Simoncelli. 2006. Noise characteristics and prior expectations in human visual speed perception. *Nature Neuroscience* 9, 4 (2006), 578–585.

[73] Niels A Taatgen. 2013. The nature and transfer of cognitive skills. *Psychological Review* 120, 3 (2013), 439.

[74] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[76] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. 2019. Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation. In *Advances in Neural Information Processing Systems*. 1–12.

[77] John H Wearden. 1991. Do humans possess an internal clock with scalar timing properties? *Learning and Motivation* 22, 1-2 (1991), 59–83.

[78] Rhiannon Weaver. 2004. Likelihood-based estimation and model selection for ACT-R cognitive models. (2004).

[79] Jacob O Wobbrock, Edward Cutrell, Susumu Harada, and I Scott MacKenzie. 2008. An error model for pointing based on Fitts' law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1613–1622.

[80] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation. In *Advances in Neural Information Processing Systems*. 14636–14647.

[81] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. 2020. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661* (2020).

[82] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*. 1094–1100.

[83] Shumin Zhai, Jing Kong, and Xiangshi Ren. 2004. Speed–accuracy tradeoff in Fitts' law tasks—on the equivalency of actual and nominal pointing precision. *International Journal of Human-Computer Studies* 61, 6 (2004), 823–856.

[84] Xiaolei Zhou. 2016. An empirical study of operational bias in steering tasks for different user groups. In *International Conference on Network and Information Systems for Computers*. 362–364.

[85] Xiaolei Zhou and Xiangshi Ren. 2010. An investigation of subjective operational biases in steering tasks evaluation. *Behaviour & Information Technology* 29, 2 (2010), 125–135.

[86] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1433–1438.

[87] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. 2019. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*. 7693–7702.