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 subprocesses. 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*

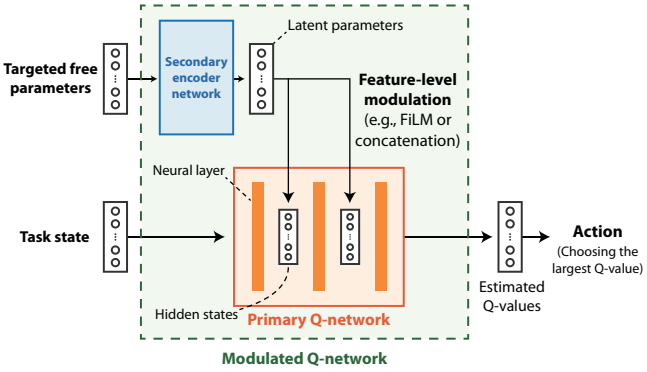

Figure 1: The structure of the modulated Q-network.

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:

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

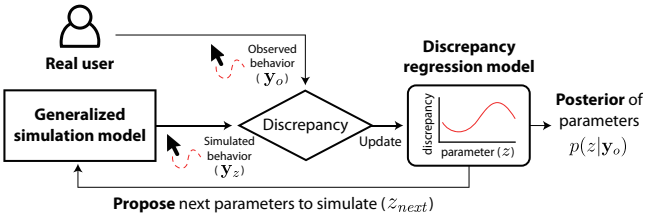

Figure 2: Overview of the BOLFI process to infer free parameters of the simulation model (z) corresponding to a real user. BOLFI constructs a regression model that estimates the discrepancy between the user's observed behavior ($\mathbf{y}_o$) and the simulated behavior of the model ($\mathbf{y}_z$) 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

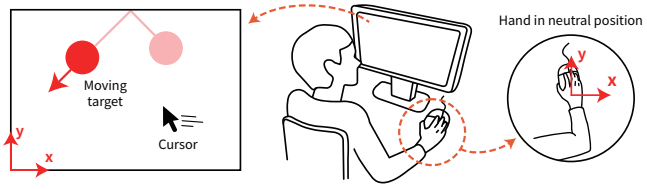

Figure 3: The point-and-click task environment covered in this study. *Center*: A right-handed user performs a point-and-click task on screen with a mouse device. *Left*: The xy coordinate of a target and a cursor on the screen. *Right*: The xy coordinate of a user's hand.

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.

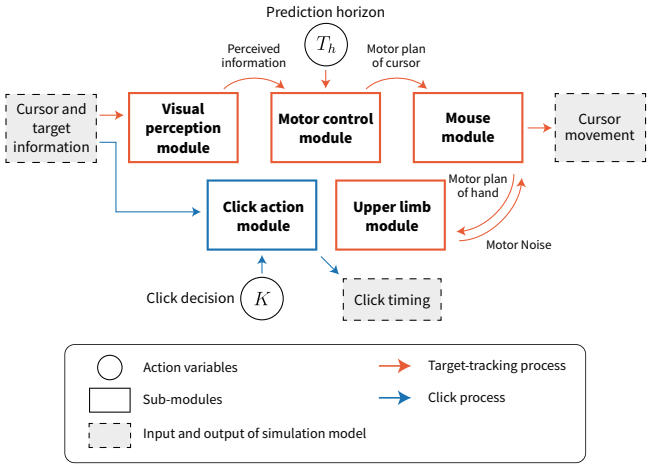

Figure 4: Overview of the point-and-click simulation model in [19]. The simulated user's point-and-click behavior (target-tracking and clicking) is implemented through a combination of five sub-modules.

4.2 Point-and-Click Process

The model describes the user's point-and-click process by introducing five sub-modules (Figure 4). First, the *motor control module* creates a motor plan that allows the cursor to reach the target's future location. At this point, a variable that determines how far into the future the user will generate the motor plan is called the *prediction horizon* T_h. Second, the *click action module* plans a timing to perform a click action to acquire the target while the cursor is moving toward the target. At this point, the module also decides whether to actually execute the planned click action, which is expressed as the value of the variable K (K=1 when decided to click, K=0 when decided to not click). Third, the *visual perception module* describes the process by which the user perceives the future position and velocity of the target and the cursor. The results of the noisy perception are passed to the motor control module and used to generate a motor plan. Fourth, the *mouse module* determines the movement of the user's hand to implement the cursor motor plan generated from the motor control module. In this module, the gain function of the mouse (implemented with `Libpointing` library [13]) and the deviation of the cursor trajectory owing to the rotation of the mouse are considered. Finally, the *upper limb module* determines the degree of rotation of the mouse and incorporates motor noise into the hand movement.

4.3 MDP Formulation

To train the simulation model through RL, the sequential decision process of the simulated user in the point-and-click task environment should be formulated as MDP, that is, in the form of states, actions, and rewards. The model defines the state and action space as follows:

- State: 11-dimensional vector with continuous values consisting of perceived information of the target and cursor, specifically, (1) perceived cursor position (2D) and velocity (2D); (2) perceived target position (2D) and velocity (2D); (3) hand position (2D); and (4) target radius (1D).
- Action: Click decision K (a binary value) and prediction horizon T_h (a discrete value ranging from 0.1 to 2.5 s with a 0.1 s interval). Therefore, the number of available actions at each state is 50 ($2\ K \times 25\ T_h$).
- Reward: Four objective terms describe an agent's task performance: (1) successful click, (2) failed click, (3) motor execution effort, and (4) elapsed time. At the agent's j-th decision-making (at timestep t=t_j), numerical values corresponding to each objective term are defined as follows:
  - Successful click: 1 if clicked successfully, otherwise 0.
  - Failed click: 1 if clicked unsuccessfully, otherwise 0.
  - Motor execution effort: sum of the absolute acceleration of the simulated hand, that is, $\sum_{t=t_j}^{t_{j+1}} \|\dot{\mathbf{v}}_h\|$, where $\dot{\mathbf{v}}_h$ is the acceleration of the simulated hand.
  - Elapsed time: time interval between the decision-making, that is, $t_{j+1} - t_j$.

  The aggregated reward r_j can be expressed as $\mathbf{w}_{rew} \cdot \mathbf{r}_j$, where $\mathbf{r}_j$ is a vector in which each component represents the numerical value for each objective, and $\mathbf{w}_{rew}$ is a vector representing the reward weight of each objective (= $[w_{success}, w_{fail}, w_{effort}, w_{time}]^T$). $w_{success}$ is set to have a positive value (compensation), whereas the other three weights have negative values (penalty). Compared to the original paper [19], we added the last time term in this paper, because we empirically found that adding the time term showed better reproduction of users' various point-and-click behaviors under different speed–accuracy bias instructions (e.g., less accurate but faster behavior). The model in [19] can be regarded as a specific case in which w_{time} is set to zero.

4.4 Free Parameters

There are 12 free parameters in the model that determine the operational characteristics of each module. They represent the cognitive and physical characteristics of a simulated user. Furthermore, if we consider the weights that determine the reward setting in the MDP formulation, the number of free parameters of the model increases to 16. Table 1 lists the symbols and meanings of all parameters, and the values they assumed in the original model [19].

Among the parameters, T_p is a constant time interval that humans spend for motor planning in the intermittent motor control process, and is a value that can be regarded as having a slight difference between users [11]. Conversely, ten of the parameters (σ_v, n_v, n_p, c_σ, c_μ, ν, δ, l_{se}, l_{ew}, and l_{ws}) can show significant differences between users. Among them, l_{se}, l_{ew}, and l_{ws} related to the geometry of the user's arm can be measured explicitly. However, the remaining parameters (σ_v, n_v, n_p, c_σ, c_μ, ν, and δ) can only be implicitly estimated by analyzing user behavior in controlled laboratory experiments. The remaining four parameters ($w_{success}$, w_{fail}, w_{effort}, and w_{time}) related to the user's intrinsic reward setting may also show differences between users. Furthermore, these parameters can change even for the same user if the context of the interaction changes (e.g., different speed–accuracy instructions given to the user) [83].

Table 1: Free parameters of the point-and-click simulation model in [19]. *Module*: Sub-module of the point-and-click model to which each parameter belongs. *Inference*: Targeted parameters to be inferred in this study.

Parameter	Meaning	Value in [19]	Module	Inference
T_p	Unit time interval for motor planning	0.1 s	Motor control	
σ_v	Visual perception noise	0.15	Visual perception	✓
n_v	Motor noise constant (parallel)	0.2	Upper limb	✓
n_p	Motor noise constant (perpendicular)	0.02	Upper limb	
l_{se}	Shoulder-to-elbow length	25.7 cm	Upper limb	
l_{ew}	Elbow-to-wrist length	25.7 cm	Upper limb	
l_{ws}	Wrist-to-hand length	6.43 cm	Upper limb	
c_σ	Precision of internal clock	0.09015	Click action	✓
c_μ	Implicit aim point	0.185	Click action	
ν	Drift rate	19.931	Click action	
δ	Visual encoding precision limit	0.399	Click action	
$f_{gain}()$	Mouse acceleration function	OS X 10.12	Mouse	
$w_{success}$	Reward weight of successful click	14	–	✓
w_{fail}	Reward weight of failed click	−1	–	✓
w_{effort}	Reward weight of motor effort	−1	–	✓
w_{time}	Reward weight of elapsed time	0	–	✓

Excluding explicitly measurable parameters and parameters that do not differ between users, seven cognitive parameters (σ_v, n_v, n_p, c_σ, c_μ, ν, and δ) and four reward parameters ($w_{success}$, w_{fail}, w_{effort}, and w_{time}) are worth inferring by analyzing the user's point-and-click behavior. Among the cognitive parameters, σ_v represents the precision with which the user perceives the speed of the target, that is, the user's visual perception performance. n_v and n_p are proportional constants that determine the amount of signal-dependent motor noise added to each of the parallel and perpendicular directions when the user wants to move the hand to the desired position. In general, n_v and n_p are expected to be highly correlated with each other (actually verified in Section 6.2.2). Finally, c_σ, c_μ, ν, and δ are variables indicating the quality of the user's click process [39, 40], and it is known that c_σ (precision of the user's internal clock) shows a significant difference between users [40, 51].

Consequently, we decided to infer the three cognitive parameters (σ_v, n_v, and c_σ) and four reward parameters ($w_{success}$, w_{fail}, w_{effort}, and w_{time}) of the model by analyzing the user's point-and-click behavior in this study, considering the importance of each parameter and the correlation between the parameters.

5 STUDY OVERVIEW

We conducted three studies (Studies 1–3) to demonstrate and evaluate our inference technique using the latest point-and-click simulation model introduced in Section 4. Specifically, we present two different generalized point-and-click simulation models based on different structures of the modulated Q-network: $\mathcal{M}_{rew}$, a simulation model generalized across the variations in *reward weights* of the simulated user (Study 2), and $\mathcal{M}_{cog}$, a simulation model generalized across the variations in *cognitive parameters* of the simulated user (Study 3). Those studies are summarized as follows:

- Study 1: We built a dataset of point-and-click behavior with 20 participants. This dataset was used in Studies 2 and 3 to evaluate the performance of our inference technique.
- Study 2: By analyzing the point-and-click behavioral data of 20 participants, we inferred the reward parameters ($w_{success}$, w_{fail}, w_{effort}, and w_{time}) of the participants, which varied for different speed–accuracy bias instructions (emphasis on speed, accuracy, or both).
- Study 3: By analyzing the point-and-click behavioral data of 20 participants, we inferred the cognitive parameters (σ_v, n_v, and c_σ) of each participant.

In the following sections, we explain the implementation and evaluation methods and discuss the results of each study.

6 STUDY 1: POINT-AND-CLICK INFERENCE DATASET

In Study 1, we built a dataset of point-and-click behaviors. This dataset is used in Studies 2 and 3 to evaluate the performance of our inference technique. The participants performed four different tasks in two days (Figure 5). Three of these tasks are to estimate baseline values of participants' cognitive parameters (σ_v, n_v, and c_σ). These tasks have been sufficiently verified in previous studies [39, 43, 72] for their significance in measuring each cognitive parameter. The other one is to measure the point-and-click behavior of participants, which will actually be used for inference in future studies.

6.1 Method

6.1.1 Participants. Twenty participants were recruited (13 women and 7 men). Among the 20 participants, 11 were under the age of 30, 5 were in their 30s, and 4 were in their 40s. Their average age was 30.4 (σ=8.65). We recruited participants from a wider age range than in previous studies [7, 19, 34, 51]. This increases the external validity of our inference study. All participants were right-handed and reported themselves familiar with the desktop environment.

6.1.2 Tasks. Participants performed four tasks in two days. Each task is described as follows.

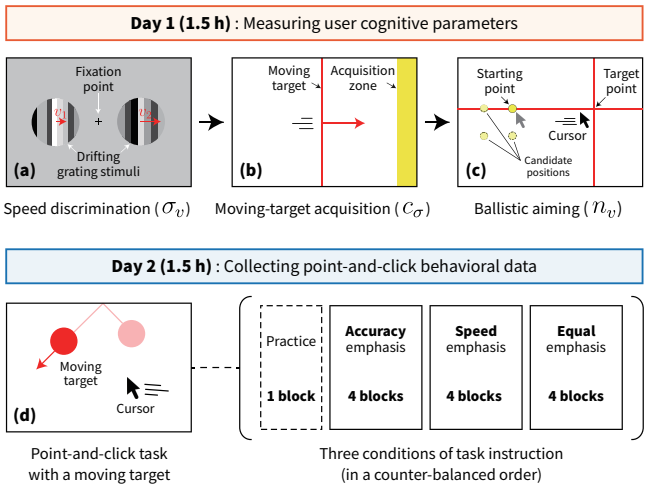

Figure 5: Overview of the laboratory experiment in Study 1. *Day 1*: Participants performed the three different tasks. Description of each experimental setup and the measured cognitive parameter are presented at each stage. *Day 2*: Participants performed a point-and-click task under three task instruction conditions (Accuracy, Speed, or Equal).

- Speed discrimination task [72]: In each trial, a pair of drifting grating stimuli were shown to the participant (Figure 5(a)). The stimulus only existed for a certain duration and then disappeared, and the participant must then answer which stimulus drifts at a higher speed (two-alternative forced choice (2AFC) paradigm). In this process, participants were asked to focus on the fixation point at the center of the screen. The speed of each stimulus can be varied from one trial to another. One of each pair of stimuli was a reference grating, and the other was a test grating, and the position of each stimulus (left or right) was randomly assigned in each trial. From this experiment, we expected to measure the baseline of the participant's visual perception noise parameter, σ_v.
- Moving-target acquisition task [39]: Participants were asked to press a button when a moving target was positioned within an acquisition zone (Figure 5(b)). The target appeared repeatedly over a specific period and was a line moving from left to right at a constant speed. Participants could not skip targets and press the button at least once in each trial. From this experiment, we expected to measure the baseline of participants' internal clock precision, c_σ.
- Ballistic aiming task [43]: Participants were asked to move a cursor from a starting point to a target point (Figure 5(c)). The trial started when the participant clicked the starting point. Subsequently, when the cursor starts moving, the starting point, target point, and cursor disappear. Consequently, the user's movement becomes ballistic. When the cursor stops moving, the trial ends and a new starting and target point are given. From this experiment, we expected to measure the baseline of participants' motor noise constant, n_v.
- Point-and-click task: Participants performed the same task as the point-and-click scenario assumed by the point-and-click simulation model (described in Section 4.1, Figure 5(d)).

6.1.3 Design. All tasks followed a full factorial within-subject design. The independent and dependent variables for each task design are described below.

- Speed discrimination task: The experiment had an independent variable, the speed of the reference grating (1, 2, 4, and 8 deg/s in visual angle). The speed of the test grating was determined by following the two interleaved adaptive staircase procedures commonly used in 2AFC tasks. The speed of the testing grating was between half and double the reference speed of the corresponding trial. The dependent variable was participant's discrimination performance (success probability) at each reference grating speed. Fifty discrimination trials were performed for each reference speed condition. The reference speed was randomly selected for each trial while controlling the total number of trials to 50.
- Moving-target acquisition task: The experiment had three independent variables (P, t_c, and W_t). P represents the period in which the target appears repeatedly and had two levels (1 or 2 s). And t_c represents the duration in which participant could observe the movement of the target that appeared in each trial, and had three levels (0, 0.08, or 0.25 s). W_t is the duration in which the target stays within the acquisition zone and has two levels (0.08 or 0.13 s). The dependent variable is the probability of participant's acquisition failure for each P-t_c-W_t combination. For each combination, 50 target acquisition trials were performed.
- Ballistic aiming task: The experiment had one independent variable, the distance from the starting to the target point (12, 32, 66, 113, 174, 248, 336, 437, or 552 pixels). To prevent participant learning, the starting point was randomly determined for each trial as one of the four candidate positions (see Figure 5(c)). For each distance condition, 40 cursor movement trials were performed. The dependent variable is the x- and y-direction standard deviation of the cursor end point distribution.
- Point-and-click task: The experiment had one independent variable, *task instruction* (Accuracy, Speed, or Equal). In the Accuracy condition, participants were instructed to click as accurately as possible. In the Speed condition, participants were instructed to click as quickly as possible. In the Equal condition, participants were asked to click as quickly and accurately as possible. To observe sufficiently diverse point-and-click behaviors from participants, we randomized the speed and radius of the target over a wide range (speed: 0–510 mm/s, radius: 9–24 mm) from one trial to another. Each participant performed four *blocks* of trials per task instruction condition, and each block consisted of 200 consecutive trials. The cursor and target trajectories were both logged.

6.1.4 Material. The captured images of the task screen are shown in Figure 6. The implementation of each task is described in more detail below.

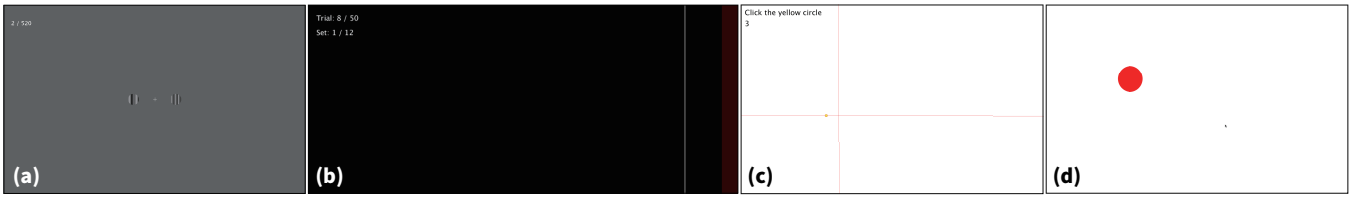

Figure 6: Captured screens of (a) speed discrimination, (b) moving-target acquisition, (c) ballistic aiming, and (d) point-and-click tasks.

- Speed discrimination task: Drifting gratings were implemented as circular patches with a diameter of 3 degrees (in visual angle). The gratings were implemented as broadband stimuli covering spatial frequencies from 1/3 cycles/deg to 2 cycles/deg (as in [72]). The phase of each frequency component was randomized and the power spectrum fell as f^{-2}. Patches were centered at 6 degrees on either side of the fixation point. The contrast[2] of the grating was set to 0.5 for both the reference and test.
- Moving-target acquisition: The speed of the target was automatically set for each condition such that it took P for the target to pass through the given screen width. After the target speed was determined, a black matte was overlaid to satisfy the t_c condition, and an acquisition zone that satisfies the W_t condition was created.
- Ballistic aiming: The candidate locations of the starting point were 100 or -100 pixels shifted along the x or y-axis from the base point (i.e., 1/3 point from the left of the screen, vertically centered). A standard cursor pointing to the upper-left side was used.
- Point-and-click task: The color of the target was red and the color of the background was white. A standard cursor pointing to the upper-left side was used.

6.1.5 Procedure. The experiment was conducted in two days to minimize the effects of fatigue on participants. On the first day, all participants were informed about the overall procedure and signed a consent form before the experiment. Participants performed the tasks in the order of speed discrimination, moving-target acquisition, and ballistic aiming. Each participant was given practice trials for each task before the measurement. Participants were given a 5-minute break between tasks. The approximate time taken for each task was as follows: speed discrimination (20 min), moving-target acquisition (15 min), and ballistic aiming (30 min). The experiment on the first day took approximately 1.5 h for each participant.

On the second day, before collecting the point-and-click behavioral data, the participants were provided a practice session of a block of trials. Subsequently, participants sequentially performed the three task instruction sets (4 blocks each) in a randomized and counterbalanced order. The experimenter verbally provided the task instructions (Accuracy, Speed, or Equal) to participants before the start of each block. Participants were given a 1-minute break between blocks and a 5-minute break between each task instruction set. Each block was performed within approximately 5 min, and the entire data collection process per participant took approximately 1.5 h for the second day.

6.1.6 Apparatus. Participants performed the experiment in a desktop environment (Mac OS Catalina 10.15) consisting of a single monitor display, keyboard, and mouse. A 24-inch (527.04 mm × 296.46 mm) monitor (Lenovo ThinkVision T24i-10) was used. The refresh rate of the display was set to 60 Hz. A wired optical mouse (Logitech G102) was used with a resolution of 1,000 DPI, a polling rate of 125 Hz, and a constant control-display gain of 10.4. The moving-target acquisition task was implemented with a size of 900 × 400 pixels, and all other task applications were run on a full-screen (1920 × 1080). All task applications were implemented in Java language and run at a frame rate of 60 Hz or higher. A TES-137 luminometer was used to measure the grating contrast in the speed discrimination task.

6.2 Results

6.2.1 Analysis. The cursor and target trajectories obtained from the point-and-click task will be used subsequently to evaluate the inference performance of our technique in Studies 2 and 3. However, before that, we performed a preliminary analysis of the participants' point-and-click performance (*success rate* and *trial completion time*) in this study. The success rate is the rate at which the participant acquired the target. The trial completion time is the time interval between the moments when the target is given and when the participant clicks. Meanwhile, we found a significant difference in trial completion time between the first block and subsequent blocks from Helmert contrast (p=0.008, Figure 7). This was considered as an effect of the participants' learning; thus, the first block was removed from all subsequent analysis and inference studies.

The goal of the remaining three tasks, except for the point-and-click task, is to estimate the baseline values of the cognitive parameters of participants. The analysis process for each task is described in detail below.

- Speed discrimination task: For each speed condition of the reference grating, we obtained the corresponding psychometric function for each participant. From this, we determined the standard deviation of the participants' speed perception distribution in each condition. This process is performed through MLE combined with the Monte Carlo method, assuming that the distribution of the participant's speed perception is Gaussian. For more details, refer to the original paper [72]. Before conducting the experiment, we performed

[2] A value obtained by dividing the maximum intensity amplitude of a grating by the maximum value of the intensity difference the monitor can display.

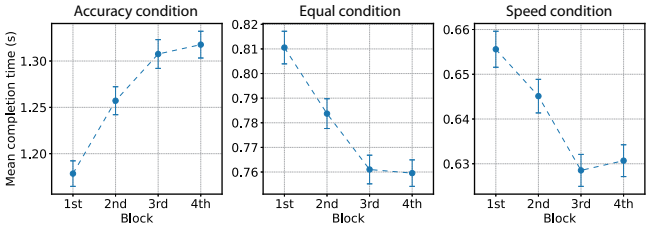

Figure 7: Mean trial completion time according to blocks in each task instruction (*left*: Accuracy, *center*: Equal, *right*: Speed). The results of 12,000 trials per block (20 participants × 3 task instructions × 200 trials) are averaged. The error bar represents one standard error of the mean.

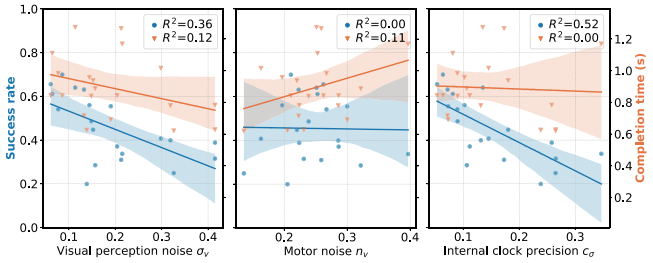

Figure 8: Point-and-click performance (success rate in *blue* and completion time in *orange*) of participants according to three cognitive parameters (*left*: σ_v, *center*: n_v, *right*: c_σ). Linear regression results are presented as solid lines along with the bands of 95% confidence interval.

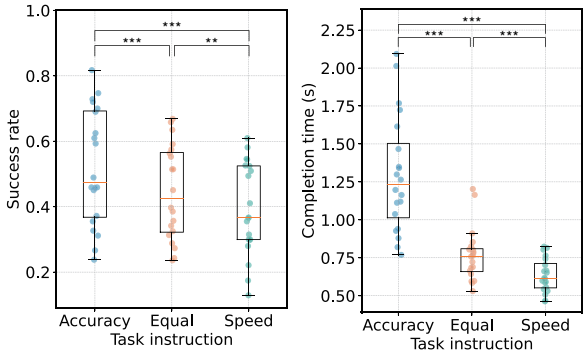

Figure 9: Box plots of the point-and-click performance (*left*: success rate, *right*: completion time) according to the task instructions. Statistically significant differences are indicated (: $p < 0.01$, ***: $p < 0.001$).**

log transformation of grating speed ($s' = \ln(1 + s/0.3)$) according to the original study [72]. Through this transformation, it can be assumed that the standard deviation (σ) of the participant's perceptual distribution is relatively constant even when the reference speed is changed. According to the original study, σ can be expressed as the product of the grating speed (s') function and the contrast (c) function: $\sigma = g(s')h(c)$. The cognitive parameter σ_v to be obtained is the average of $g(s')$ for each participant. We approximated $h(c)$ to be 2.0, the average value observed in the original study, and estimated $g(s')$ values from σ.

- Moving-target acquisition task: According to the model proposed by the previous study [39], for each P-t_c-W_t condition, we can express the participant's error rate E as follows:

$$E = 1 - \frac{1}{2}\left[erf\left(\frac{W_t - \mu}{\sigma\sqrt{2}}\right) + erf\left(\frac{\mu}{\sigma\sqrt{2}}\right)\right],$$

$$\text{where } \mu = c_\mu \cdot W_t$$

$$\text{and } \sigma = \frac{c_\sigma \cdot P}{\sqrt{1 + (P/(1/(e^{\nu t_c} - 1) + \delta))^2}}.$$

  By fitting the empirically observed error rate to the above equation, we can estimate the cognitive parameters (c_σ, c_μ, ν, and δ) for each participant.
- Ballistic aiming task: We first simulate the users' ballistic aiming movement with various n_v-n_p sets, using the point-and-click simulation model (Section 4). Through this, we can simulate the end point distributions of the ballistic movement for each n_v-n_p set, and quantitatively obtain the *slope* of standard deviations of the distributions according to target distances. We determine the n_v and n_p, which minimize the discrepancy between the slope of the participant and that of simulation, and set them as the measured values for each participant. The simulation was set up under the same conditions as the task performed by the participants (for more details of the simulation, refer to the previous study [19]).

6.2.2 Cognitive parameters. We obtained the participants' baseline cognitive parameters as follows: σ_v (μ=0.169, σ=0.082), c_σ (μ=0.149, σ=0.084), c_μ (μ=0.385, σ=0.142), ν (μ=15.766, σ=5.376), δ (μ=7.81e−3, σ=1.43e−2), n_v (μ=0.245, σ=0.056), and n_p (μ=0.047, σ=0.033). Among the measured σ_v values, we observed that the two extremely high values (0.730 and 1.191) induced the instability of optimization of the point-and-click simulation model in Section 4. Specifically, owing to the extremely high visual perception noise, the simulated user's perceived information of the target and cursor varied significantly, which prevented convergence of the simulated user's policy. This problem of needing an appropriate upper bound for σ_v is our new finding that has not been reported in the original paper [19]. To enable stable optimization of the simulation model, we adjusted the σ_v values of two outliers to the upper three-sigma value (=0.415). As mentioned in Section 4.4, a significant correlation existed between n_v and n_p (Pearson's r=0.460, p=0.041). Therefore, in subsequent studies, we assumed a linear relationship between the n_v and n_p values, according to their measured mean values ($n_p = 0.192 \times n_v$). We also examined the correlation between the three targeted cognitive parameters (σ_v, c_σ, and n_v). There was a significant correlation between σ_v and c_σ (Pearson's r=0.643, p=0.002); however, no other significant correlations were found (p>0.05).

6.2.3 Point-and-click performance. The average success rate of all participants was 45.4%, and the trial completion time was 0.89 s

(σ=0.20 s). Figure 8 shows the point-and-click performance of individual participants according to their measured cognitive parameters. We examined the correlation between participants' cognitive parameters and their performance. Both σ_v and c_σ exhibited a high correlation with the success rate (for σ_v, Pearson's r=−0.603, p=0.005; for c_σ, Pearson's r=−0.721, p<0.001). No other significant correlations were found.

Figure 9 shows how the task instruction changed the participants' behavioral strategies (levels of speed–accuracy trade-off). The average performance of the participants according to each task instruction was as follows: for Accuracy, success rate=52.1%, completion time=1.295 s (σ=0.383 s); for Equal, success rate=44.4%, completion time=0.768 s (σ=0.173 s); for Speed, success rate=39.2%, completion time=0.635 s (σ=0.112 s). A repeated measure ANOVA revealed that there were significant effects of the task instruction on both performance metrics (for success rate, $F_{2,38}$=76.56, p<0.001; for completion time, $F_{2,38}$=24.62, p<0.001). We indicated all the post hoc test (pairwise t-test) results between the conditions in Figure 9.

6.3 Discussion

In Study 1, we built a dataset of point-and-click behaviors from 20 participants through a controlled laboratory experiment. Compared to the latest point-and-click dataset (mean success rate=62.3%, mean completion time=0.89 s) [51], the participants in our dataset exhibited a lower mean success rate (45.4%) and the same mean completion time (0.89 s). This was expected because the demographic composition of participants changed and, in particular, the participants' average age increased (25.02 to 30.40). In addition, the change in the mouse setting (e.g., control-display gain) could contribute to the lower mean success rate.

6.3.1 Baseline values of cognitive parameters. We obtained baseline values of the cognitive parameters from each of the 20 participants. This allows us to reliably determine the free parameters of the simulation model than simply applying typical mean values reported in previous studies. Compared to the previous point-and-click study [19], which employed the literature value (see Table 1), the mean baseline values of the three target cognitive parameters were different as follows: σ_v (increased from 0.15 to 0.169), n_v (increased from 0.2 to 0.245), and c_σ (increased from 0.090 to 0.169). The distributions of measured values include the literature values within the one-sigma range (i.e., $\mu \pm \sigma$).

6.3.2 Correlations between cognitive parameters. The correlation between n_v and n_p was expected as it is a general observation. In addition, we found an unexpected correlation between σ_v and c_σ, two of the three target cognitive parameters for inference. Both σ_v and c_σ represent participant's ability to perceive a specific visual stimulus given for a short period. In this perception process, participants have to encode visual stimuli in their working memory (or visual image store) within a short time [12]; thus, the performance of both speed perception [22], which is related to σ_v, and timing perception [16], which is related to c_σ, can be commonly influenced by the capacity of participant's working memory. If the correlation between those target parameters was not discovered, the inference (BOLFI) would be performed using an incorrect prior distribution of the target parameters, thereby leading to an incorrect posterior estimation (i.e., degrading the inference performance). Therefore, the correlation between σ_v and c_σ was an important observation and it was considered in the inference process in Study 3 (Section 8.2).

6.3.3 Effects of cognitive parameters. The participants' success ratios on the point-and-click trials were influenced by two of the three measured cognitive parameters (σ_v and c_σ). We found that a participant with a lower visual perception noise (lower σ_v) or a more precise internal clock (lower c_σ) performed the point-and-click task more accurately. To successfully click on the target, it is necessary to accurately estimate *both* the relative speed between the target and the cursor and the timing at which the cursor is positioned within the target; thus, this is an expected result. However, n_v did not demonstrate a correlation with any task performance in our experiments. The n_v variations between the participants may not be sufficiently diverse to cause a significant difference in the point-and-click behaviors. In the future, a group of users with more special motor characteristics may be included (e.g., seniors or kids).

6.3.4 Effect of task instruction. When given different task instructions, the participants clearly exhibited different task performances; a significant difference was observed in every combination of the pairwise t-test (Figure 9). A clear and well-known trade-off relationship was observed; the participants could perform the tasks more precisely as needed by spending more time. Our results accurately replicated the results of previous studies [44, 83], reporting that a user can flexibly change their point-and-click strategy according to a given task instruction.

7 STUDY 2: INFERRING REWARD WEIGHTS

In Study 2, we infer the reward weights ($w_{success}$, w_{fail}, w_{effort}, and w_{time}) of participants. This is equivalent to identifying the intrinsic reward settings of users to perform a point-and-click task. To achieve this, we first train $\mathcal{M}_{rew}$, a generalized simulation model over variations in *reward* weights. The model is implemented with an action policy based on our modulated Q-network that uses the reward weights as targeted free parameters to generalize. Consequently, the optimized simulation model ($\mathcal{M}_{rew}$) can immediately adapt its behavior according to given reward weights. We demonstrate that, using $\mathcal{M}_{rew}$, the intrinsic reward settings of users can be inferred with improved efficiency.

7.1 Model Training

In order to train $\mathcal{M}_{rew}$, we applied the envelope MOQL [80], which is one of the most recent multi-objective RL algorithms. For an agent behaving in a task environment with several objective terms, the algorithm provides a specialized method of training the action policy that can respond according to the given reward weights ($\mathbf{w}_{rew}$) using a form of multi-objective Q-network. We describe the details of the method of incorporating the modulated Q-network into the algorithm.

7.1.1 Policy model architecture. We implemented the policy model of $\mathcal{M}_{rew}$ based on our modulated Q-network, and modified the output part to have the form of a multi-objective Q-network [80] (Figure 10). The core difference between the multi-objective and general (single-objective) Q-networks is the use of a *vectorized* form

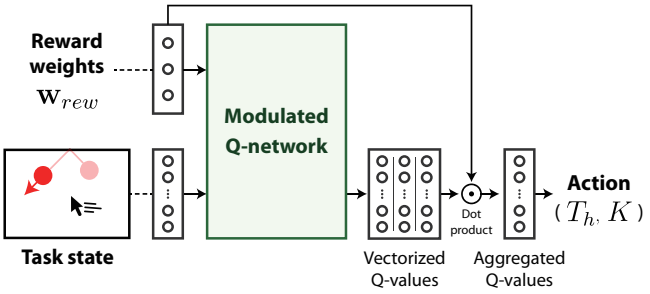

Figure 10: The structure of the policy model of the generalized simulation model over reward weights ($\mathcal{M}_{rew}$).

of Q-values. While single-objective Q-networks predict a *scalarized* Q-value for each state-action pair, a multi-objective Q-network predicts a vectorized Q-value for each state-action pair, consisting of a Q-value corresponding to each objective term. In the multi-objective case, the vectorized Q-values are dot-produced by the reward weight vector ($\mathbf{w}_{rew}$) to form an aggregated Q-value. Thus, the optimal behavior of a trained agent can be achieved by selecting the action with the highest aggregated Q-value.

We set the modulated Q-network to receive the target reward weight vector ($\mathbf{w}_{rew}$) as an auxiliary input along with the task state, and to output the vectorized Q-values. The primary Q-network (within the modulated Q-network) consisted of three fully connected (FC) neural layers. The first two layers consisted of 64 hidden units with ReLU activation. The output layer had a size of 200, determined by the product of the action dimension (=50) and the number of objective terms (=4). We used feature-level concatenation, which empirically exhibited better modulation performance than the FiLM method in this setting; however, FiLM can be a better modulation method for simulation models in other task environments. The secondary encoder network consisted of three FC layers. The first two layers consisted of 16 hidden units with ReLU activation. The feature size of the last layer (i.e., the size of the latent parameters) was 16. We allowed the concatenation of the latent parameters only in the first hidden states (i.e., outputs of the first layer). We used normalized values of the reward weights (values mapped between -1 and 1) when they were fed as inputs of the modulated Q-network.

7.1.2 Training details. We followed the envelope MOQL algorithm to train the policy model (pseudo-code in [80]). The key differences between the envelope MOQL algorithm and the original DQN are as follows: (1) the vectorized rewards and Q-values are used; (2) the target reward weights are randomly sampled for every episode; therefore, the agent explores over various reward settings; (3) a specialized loss function (enabling *homotopy* optimization) is used for model updating; (4) the training data (i.e., experiences) for the model update are augmented by applying multiple different reward weights to a single MDP transition.

During the training phase, we fixed w_{fail} to -1 and sampled the remaining three reward weights within the following ranges: $[2.5, 40]$ for $w_{success}$; $[-8, -0.5]$ for w_{effort}; and $[-12, -0.75]$ for w_{time}. Because the optimal action (with the highest aggregated Q-value) is determined by the ratio of the reward weights, we could indirectly investigate the effect of w_{fail} by adjusting the remaining three reward weights. The cognitive parameters of the simulated user were set to the average values of the 20 participants measured in Study 1. The mouse acceleration function (f_{gain}) was set to the same as used in the experiment in Study 1 (with fixed control-display gain), and other remaining free parameters (T_p, l_{se}, l_{ew}, and l_{ws}) were set to the values of the previous study [19]. We trained the model for total 1M training steps and the entire training process took 4–5 days (using `Intel` Xeon E5-2630v4 CPU, 2.2 GHz). An Adam optimizer with the inverse-square-root learning rate schedule [75] (learning rate=0.001, warm-up step=5K) was used. The replay memory size was 100K, the batch size was 1024, and the discount factor (γ) was 0.95.

7.2 Inference

With the trained $\mathcal{M}_{rew}$, we applied BOLFI to fit (i.e., infer) the reward weights ($w_{success}$, w_{effort}, and w_{time}), for each of the three task instructions, that best describe the participants' behavioral data collected in Study 1. For each task instruction, there were 12,000 trials of behavioral data from the 20 participants, and we used 2,400 trials for the inference. Accordingly, during BOLFI, simulation of 2,400 trials, under the same conditions as performed by participants (the same initial cursor position, target position, velocity, and radius), was conducted per sample acquisition process. We binned the 2,400 trials using the target speed and radius (four equal-frequency bins for each). Using eight bins, the discrepancy function was defined as $\sum_{bins}((SR_{obs} - SR_{sim})^2 + a \times (CT_{obs} - CT_{sim})^2)$, where SR and CT denote the average success rate and completion time in each bin (the subscripts *obs* and *sim* represent observed and simulated data), respectively, and a is the coefficient that balances the two metrics. BOLFI was performed with 100 sample acquisition processes for each task instruction. Each inference process (i.e., simulation of 2,400 trials $\times$ 100 samples) took approximately four hours.

7.3 Evaluation and Results

7.3.1 Generalization performance. To evaluate the generalization performance of our trained $\mathcal{M}_{rew}$, we investigated the simulated behaviors of $\mathcal{M}_{rew}$ adapted to different sets of reward weights. We individually trained simulation models with the same sets of fixed reward weights (one individually trained model per fixed reward weight set), and compared the simulated behaviors of the individually trained models and our $\mathcal{M}_{rew}$ that was modulated with the corresponding reward weight set. If $\mathcal{M}_{rew}$ exhibits simulated behavior similar to that of each individually trained model for each reward weight set, the generalization performance of $\mathcal{M}_{rew}$ can be judged as satisfactory.

For this evaluation, we prepared seven sets of reward weights (i.e., corresponding to seven hypothetical users with different intrinsic reward settings); one set had the intermediate (*mid*) values for the three reward weights ($w_{success}$, w_{effort}, and w_{time}); the other six sets had one of the three reward weights *high* or *low* (2×3 combinations). We set the (*high*, *mid*, and *low*) values of each reward weight to have the same distance from the logarithmic scale as follows: $w_{success}$ (25.0, 10.0, 4.0), w_{effort} (-5.0, -2.0, -0.8), and w_{time} (-7.5, -3.0, -1.2). For each of the seven sets, one individually trained model was trained based on the same setting with $\mathcal{M}_{rew}$ (each model took

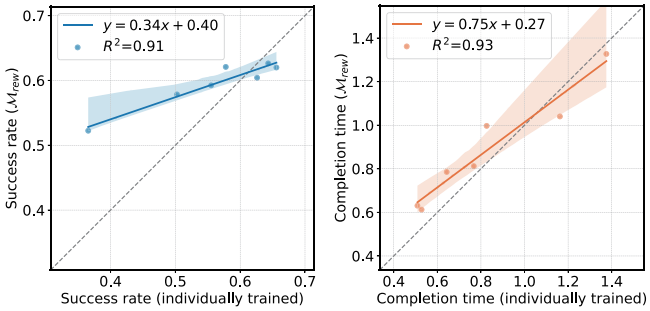

Figure 11: Simulated point-and-click performance (*left*: success rate, *right*: completion time) of the individually trained models (x-axis) and our $\mathcal{M}_{rew}$ (y-axis). The simulated results of the two models are compared under 7 different reward weight sets. Linear regression results are presented as solid lines along with the bands of 95% confidence interval.

approximately 1.5–2.5 days for training). The key differences in the individually trained model compared with $\mathcal{M}_{rew}$ were (1) the absence of policy modulation and (2) fixed reward weights during the training phase.

We compared the simulated behavior of the models in two aspects: (1) the simulated task performance and (2) the simulated user's prediction horizon (T_h) trend within a trial. For the comparison, we simulated 9,000 trials for the individually trained models and $\mathcal{M}_{rew}$, under the same initial conditions. Figure 11 shows the comparison of the simulated performance of both models. Our model ($\mathcal{M}_{rew}$) reasonably reproduced the task performance of the individually trained models with high coefficients of determination (success rate: R^2=0.91, completion time: R^2=0.93). We also compared the simulated user's T_h, which is determined solely by the action policy, and thus confirmed whether the action policy of $\mathcal{M}_{rew}$ has actually adapted. Figure 12 visualizes the change of T_h in a trial according to the variations in the values of each reward weight, for the individually trained cases and $\mathcal{M}_{rew}$, respectively. $\mathcal{M}_{rew}$ clearly reproduced the variation in T_h shown in the individually trained models, especially, in terms of the increase or decrease in T_h according to the variations in each value of the reward weight.

7.3.2 Inference performance. Figure 13 shows the inferred reward weights for each task instruction. For inference, we used the MAP values of the posterior distribution obtained through BOLFI. The exact values of inferred reward weights were as follows: $w_{success}$=6.08, w_{effort}=−4.46, w_{time}=−3.19, for Accuracy; $w_{success}$=3.10, w_{effort}=−4.19, w_{time}=−11.82, for Equal; $w_{success}$=2.50, w_{effort}=−2.64, w_{time}=−12.00, for Speed. These values represent the reward formulations for each task instruction in which average participants are expected to have; that is, the reward settings are inferred at the population level.

7.4 Discussion

In Study 2, we validated our method to implement the generalized model $\mathcal{M}_{rew}$. The trained $\mathcal{M}_{rew}$ exhibited satisfactory generalization performance in that the model successfully reproduced the simulated behaviors of all individually trained models with different sets of reward weights. While the previous point-and-click model [19] employed a hand-tuned reward formulation, we demonstrated that our $\mathcal{M}_{rew}$ can provide an automated method to infer the reward formulation from the observed behaviors of participants.

7.4.1 Effects of reward weight variation. An advantage of the generalized model is that, as shown in Figure 12, we can easily investigate the change in the simulated behavior according to the variations in each reward weight, without the need to train a separate model with the changed reward formulation. We confirmed that each of the three reward weights ($w_{success}$, w_{effort}, and w_{time}) influenced the simulated behavioral strategy (i.e., the prediction horizon during a trial) as follows: the increase in $w_{success}$ and w_{time}, and the decrease in w_{effort} led to the decrease in T_h on average. The shorter T_h indicates that the simulated user assumes the strategy of tracking the moving target by spending motor effort, rather than waiting for the target to approach them. From this viewpoint, the change in T_h in Figure 12 can be interpreted as follows. The users increase their T_h when they intend to lower their effort (under high $|w_{effort}|$). Conversely, the users spend more effort by decreasing T_h when they need to perform the task more accurately (i.e., high $|w_{success}|$) or faster (i.e., high $|w_{time}|$).

7.4.2 Computational efficiency. A noteworthy contribution of our inference method is its enhanced computational efficiency. Our method exhibited a significant reduction in the computational cost for the inverse modeling of HCI models. For example, the inverse modeling of the menu-search model [34] reported that one parameter acquisition took six hours, mainly because of the model training period. The model training period of the point-and-click model was even longer than that of this previous study. For example, the training process of each individually trained model took approximately two days. However, our inference method took a few minutes for one parameter acquisition, because no training period was required as the model could be adapted to the cognitive parameters immediately. With our method, the reward weight inference process took only four hours per task instruction.

7.4.3 Fitted reward weights. Our estimated reward weights can provide a plausible explanation for the change in the participants' behavioral strategy according to the given task instructions. Following the results in Figure 13, we can interpret that the participants decreased their intrinsic compensation for a successful click (i.e., lower $|w_{success}|$) and increased their intrinsic penalty according to the time spent (i.e., higher $|w_{time}|$), when receiving the Speed task instruction compared to the Accuracy task instruction. In addition, the magnitude of w_{effort} was decreased, which indicates that participants were willing to execute more motor effort to increase their speed. These changes in reward weights matched with the participants' less accurate but faster task performance in the Speed task instruction (Figure 9). In the Equal task instruction, all three reward weights were fitted in the middle between the Accuracy and Speed task instructions. We conclude that our method can reasonably estimate the participants' intrinsic reward formulation, in that it provides a plausible explanation for participants' behaviors at the different levels of speed–accuracy trade-off. In Study 3, the fitted

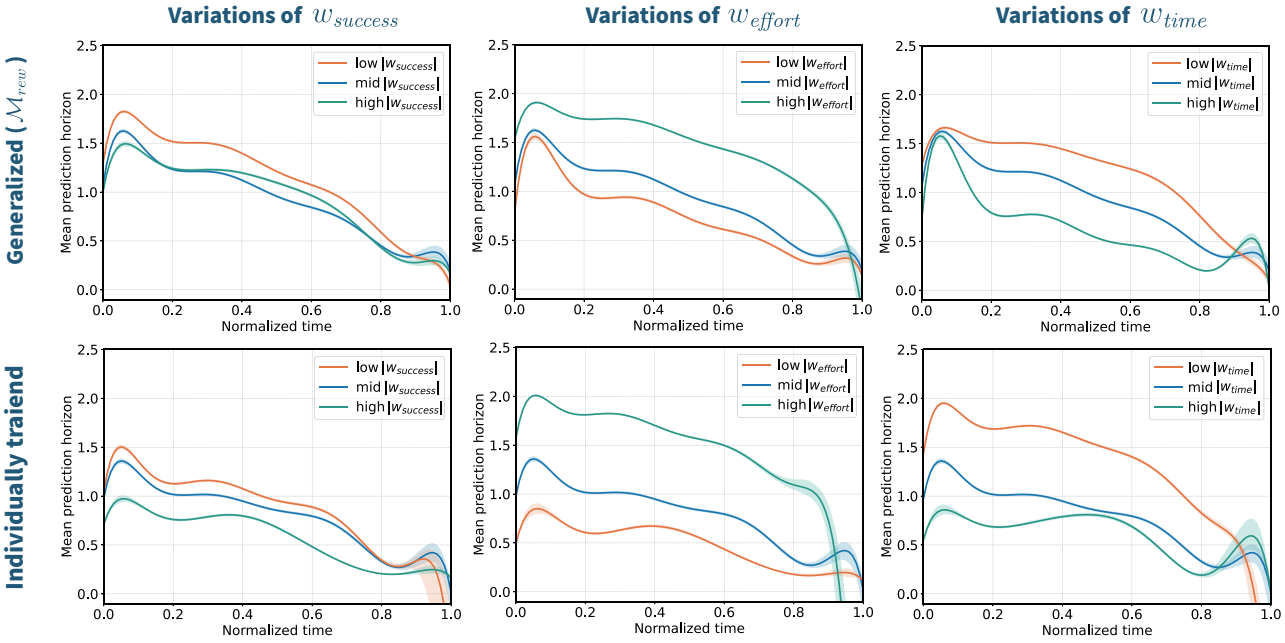

Figure 12: Mean prediction horizon over normalized time of $\mathcal{M}_{rew}$ (*top*) and individually trained models (*bottom*). We investigated the changes in prediction horizon with variations in the absolute value of each reward weight (*orange*: low, *blue*: middle, *green*: high). Each column represents the varying reward weight (*left*: $w_{success}$, *center*: w_{effort}, *right*: w_{time}). Each line (mean prediction horizon over time) was obtained by averaging the results from simulations of 9,000 trials.

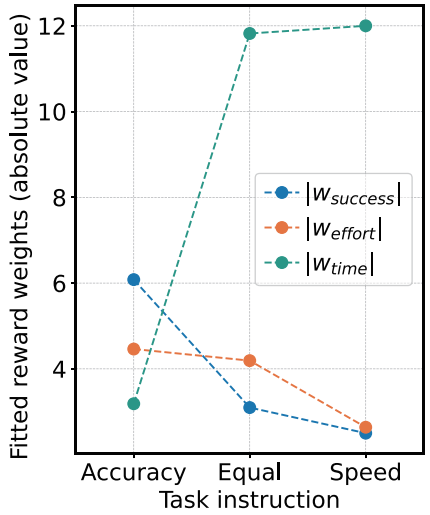

Figure 13: Results of the reward weights fitted for each task instruction in Study 2. The absolute value of each weight is shown; $w_{success}$ is positive (compensation); w_{effort} and w_{time} are negative (penalty). w_{fail} was set to -1.

reward formulations are used to train a simulation model that generalizes to different cognitive parameters of individual participants.

7.4.4 Inverse reinforcement learning. Inverse RL algorithms (e.g., [48, 86]) may be considered another possible approach to infer users' reward formulation. Inverse RL infers the reward formulation of expert agents based on their demonstrations. However, as argued by Kangasrääsiö *et al.* [34], the previous inverse RL algorithms cannot be readily used in user simulators in the field of HCI. This is because the methods are usually required to observe the environmental states and actions of the expert agents. In user simulator cases, the states and actions often refer to the inherent decision processes of humans, which are not observable. For example, in our point-and-click case, we can access the cursor trajectories performed by participants; however, we cannot identify the actual velocity and position of the target perceived (task state) or the prediction horizon of the motor plan set by the participants (action). Instead, we demonstrated that simulation-based inference methods (e.g., BOLFI) can be a suitable solution. Because the methods operate based on the evaluation of the discrepancy between the observable behaviors (e.g., cursor trajectory or task performance) of real and simulated users, the inference is available without requiring access to a user's inherent decision process.

8 STUDY 3: INFERRING COGNITIVE PARAMETERS

In Study 3, we infer the cognitive parameters (σ_v, n_v, and c_σ) of each of the 20 participants using the proposed framework in Section 3. Accordingly, we implement $\mathcal{M}_{cog}$, a generalized simulation model over *cognitive* parameters, equipped with an action policy based on our modulated Q-network. $\mathcal{M}_{cog}$ uses the cognitive parameters as the targeted free parameters to generalize, whereas

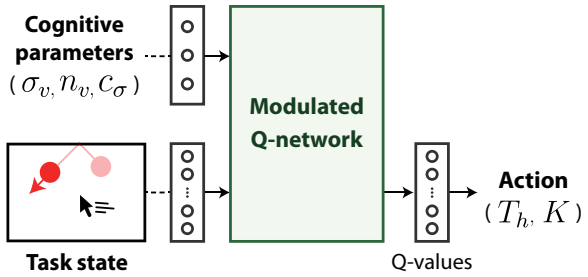

Figure 14: The structure of the policy model of the generalized simulation model over cognitive parameters ($\mathcal{M}_{cog}$).

$\mathcal{M}_{rew}$ in Study 2 sets the reward weights as the targeted free parameters. We verify the generalization performance of the trained $\mathcal{M}_{cog}$, along with the individual-level inference performance.

8.1 Model Training

8.1.1 Policy model architecture. We implemented the policy model of $\mathcal{M}_{cog}$ based on our modulated Q-network (Figure 14). The modulated Q-network receives the target cognitive parameters (σ_v, n_v, and c_σ) as auxiliary inputs along with the task state, and predicts Q-values. The primary Q-network consisted of three FC layers. The first two layers consisted of 64 hidden units with ReLU activation. The feature size of the last layer was set as our action dimension (=50). For the modulation method, we used feature-level concatenation, which empirically exhibited better modulation performance than the FiLM method in our setting. In addition, better learning performance was exhibited by skipping a secondary encoder network and directly concatenating the cognitive parameter vector to the hidden states of the primary Q-network. We used normalized values (between −1 and 1) of each cognitive parameter as inputs to the secondary encoder network.

8.1.2 Training details. We first defined an appropriate range of each cognitive parameter for generalization (the upper and lower bounds of each parameter that $\mathcal{M}_{cog}$ can simulate). The range was set according to the mean and standard deviation of the participants' baseline value distribution measured in Study 1. The cognitive parameters have positive values, and they commonly show a skewed distribution towards zero. Here, if we set the range according to the common three-sigma bounds in the linear scale, the lower bound is likely to be below zero, which is not realistic and makes the training difficult. Therefore, we set the upper and lower bounds at the same distance on a *logarithmic* scale from participants' average values. The skewness of the distribution was not evident on a logarithmic scale. The distance to each bound from the average value was determined as the upper three-sigma value on a linear scale. Accordingly, the defined ranges of the three targeted cognitive parameters were as follows: [0.069, 0.415] for σ_v, [0.145, 0.413] for n_v, and [0.055, 0.400] for c_σ.

Based on the three fitted reward formulations (in Study 2) for the three task instructions, we trained three separate $\mathcal{M}_{cog}$ corresponding to each task instruction. We applied double DQN [74], a DQN family known to exhibit more stable learning than DQN by preventing a Q-network from overestimating Q-values. The major difference between the two algorithms is the method of calculating the TD error; therefore, we could apply the training method in Section 3.2.2, without further modification. Other than the three target cognitive parameters (σ_v, n_v, and c_σ), the remaining cognitive parameters (c_μ, ν, and δ) of the simulated user were set to the average values measured in Study 1 (except for n_p, which maintains a linear relationship with n_v). We trained the model using 1.5M training steps (approximately 2–3 days). For the remainder, the same hyperparameters were used as in Section 7.1.2.

8.2 Inference

The BOLFI was conducted individually for each observed behavioral data of the 20 participants. For the inference, we employed 900 trials of behavioral data from each participant, consisting of 300 trials for each task instruction. Accordingly, 900 trials under the same task conditions were simulated for each sample acquisition process. Three separate $\mathcal{M}_{cog}$ learned for each task instruction were used to simulate the corresponding 300 trials each.

The targeted cognitive parameters influence different stages of a user's point-and-behavior: σ_v (visual perception), n_v (cursor movement), and c_σ (estimation of click timing). Therefore, to properly infer all the cognitive parameters, it is reasonable to devise a discrepancy function that considers the full cursor trajectory that the participant exhibits until a click. We defined the discrepancy function of observed and simulated data as $\sum_{trials}(d_{success} + b \times d_{traj})$ where $d_{success}$ represents the discrepancy for click results, that is, 1 if the click results (success or failure) of the observed and simulated trials do not match, otherwise 0; d_{traj} represents the discrepancy for trajectory, that is, MSE of the cursor positions on the observed and simulated trajectories at 0.05 s intervals; and b is the coefficient that balances the two discrepancy terms.

Because we observed a high correlation between σ_v and c_σ in Study 1, we reflected this relationship in the inference process. Accordingly, we obtained a linear function fitted by the baseline values of σ_v and c_σ ($c_\sigma = 0.497 \times \sigma_v + 0.053$). During the inference process, we assumed that c_σ is in the range of ± 1 standard deviation (=0.084) from the predicted c_σ value by the linear function for a given σ_v. Accordingly, BOLFI determined the following three values within the corresponding ranges: (1) σ_v (ranging from 0.069 to 0.415, as defined in Section 8.1.2), (2) n_v (ranging from 0.145 to 0.413), and (3) the difference between c_σ and the predicted c_σ using the linear function and the inferred σ_v (ranging from −0.084 to 0.084).

The inference of each participant's cognitive parameters was performed using the 100 sample acquisition processes of BOLFI. Approximately 1.5 h were spent to infer the cognitive parameters of each individual user; 30 h of CPU time was spent to infer the cognitive parameters of all 20 participants.

8.3 Evaluation and Results

8.3.1 Generalization performance. Similarly, we evaluated the generalization performance of the trained $\mathcal{M}_{cog}$ as in Study 2. We compared $\mathcal{M}_{cog}$ and individually trained models with given fixed sets of cognitive parameters in two aspects: (1) the simulated task performance and (2) the prediction horizon (T_h) within a trial. Seven fixed sets of cognitive parameters were used; one set had the average values for all three target parameters (σ_v, n_v, and c_σ); the other

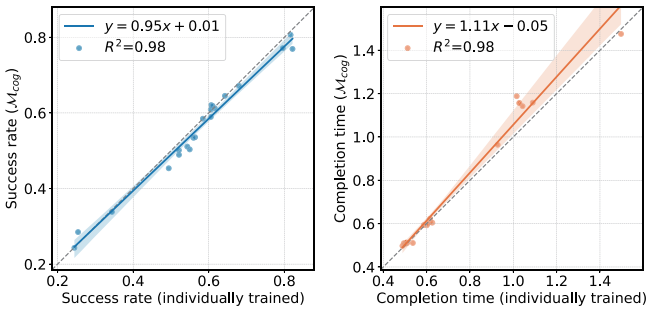

Figure 15: Simulated point-and-click performance (*left*: success rate, *right*: completion time) of the individually trained models (x-axis) and our $\mathcal{M}_{cog}$ (y-axis). The simulated results of the two are compared under 21 different settings (3 task instructions $\times$ 7 cognitive parameter sets). Linear regression results are presented as solid lines along with the bands of 95% confidence interval.

six sets had one of the three parameters *high* (the two-third point from the mean to the upper bound) or *low* (the two-third point to the lower bound). Because we implemented $\mathcal{M}_{cog}$ for each of the three task instructions, 21 models (3 task instructions $\times$ 7 cognitive parameter sets) were individually trained. For the comparison, we simulated 12,000 trials for both the individually trained models and $\mathcal{M}_{cog}$, under the same initial conditions.

As shown in Figure 15, our model ($\mathcal{M}_{cog}$) successfully reproduced the task performance of the individually trained models with high coefficients of determination (success rate: R^2=0.98, completion time: R^2=0.98). Figure 16 exhibits the change of T_h in a trial according to the variations in each cognitive parameter value, for the individually trained cases and $\mathcal{M}_{cog}$, respectively. It is observed that $\mathcal{M}_{cog}$ exhibits very similar results to the T_h change of the individually trained models. For example, the individually trained cases confirmed that the simulated user's T_h showed large variations according to the change in σ_v compared to the other two cognitive parameters, and it was reproduced by $\mathcal{M}_{cog}$.

8.3.2 Inference performance. Our inference performance was evaluated based on the baseline cognitive parameters of each individual user measured in Study 1. Figure 17 shows the correlation between the inferred cognitive parameters of each individual user and the baseline value of that user. The results show that our method predicts σ_v and c_σ of individual users with a moderate level of coefficient of determination (R^2=0.50 for σ_v, R^2=0.62 for c_σ). For the motor noise n_v, a sufficient correlation between the predicted and baseline values was not obtained (R^2=0.01).

8.4 Discussion

In Study 3, we validated our proposed method of implementing a generalized model $\mathcal{M}_{cog}$ over variations in cognitive parameters. With the trained $\mathcal{M}_{cog}$, we reasonably inferred two cognitive parameters (σ_v and c_σ) from an individual participant's observed behavior. Such inference performance was achieved based on the simulation model ($\mathcal{M}_{cog}$), which was trained only in the virtual RL environment (i.e., *in silico*). $\mathcal{M}_{cog}$ was not provided any information on the relationships between the baseline cognitive parameters and behavioral data of the participants during the training period.

8.4.1 Generalization performance. The trained $\mathcal{M}_{cog}$ exhibited excellent generalization performance, successfully approximating individually trained models with different sets of cognitive parameters. In particular, $\mathcal{M}_{cog}$ achieved even better similarity to the simulated performance of the individually trained models (Figure 15), than $\mathcal{M}_{rew}$ in Study 2 (Figure 11). The variations in cognitive parameters (in Study 3) influence the operation of each sub-module within $\mathcal{M}_{cog}$; for example, σ_v is involved in the visual perception module (Table 1). Conversely, the variations in reward weights (in Study 2) do not influence the sub-modules within $\mathcal{M}_{rew}$. Accordingly, the simulated performances of $\mathcal{M}_{cog}$ in Figure 15 were actually influenced by not only the adapted action policy but also the changed operation of the sub-modules. That is, the effect of the action policy on the simulated task performance in the case of $\mathcal{M}_{cog}$ was less than that of $\mathcal{M}_{rew}$; therefore, the better similarity could be observed in Figure 15.

Meanwhile, the simulated user's T_h is determined solely by the action policy; therefore, the comparison of T_h can demonstrate whether the action policy was successfully adapted. We confirmed that both $\mathcal{M}_{rew}$ and $\mathcal{M}_{cog}$ faithfully reproduced the trend of T_h of the individually trained models (Figure 12 and Figure 16).

8.4.2 Effect of cognitive parameter variation. As shown in Figure 16, we can investigate the effect of the cognitive parameter variations on the simulated behavior using our generalized simulation model. $\mathcal{M}_{cog}$ confirmed a clear change in the trend of simulated user's T_h according to the variations in σ_v. Higher σ_v led to a longer T_h over the entire trial. Do *et al.* [19] explained that the simulated user takes a strategy to increase T_h to wait for the target bouncing off and reduce their movement effort when the given trial is perceived to be difficult (target is small or moves fast). Similarly, we can interpret that the simulated user with a higher σ_v takes the strategy of waiting more frequently and keeping a longer T_h.

The other two cognitive parameters (c_σ and n_v) were found by the generalized model $\mathcal{M}_{cog}$ to have less effect on the overall trend of T_h, compared to σ_v. It is expected that c_σ influences the behavior of the simulated user only at the end of each trial because c_σ is involved only in a user's click timing estimation, which is the fnal step of a trial. In Figure 16, the simulated user with a lower c_σ in both $\mathcal{M}_{cog}$ and individually trained cases rapidly decreased T_h at the end of the trial. In the case of n_v, n_v may not be a factor that influences the simulated user's prediction horizon; or the range of n_v measured from the participants may not be sufficiently large to lead to a difference in the action policy.

8.4.3 Computational efficiency. Simulation-based inference at the individual level was almost infeasible in previous studies, owing to the prohibitively high computational cost. However, with the enhanced time efficiency of our proposed inference method, the individual-level inference for many users becomes practical for the first time in user simulator studies. In Study 3, it took less than one minute for one sample acquisition (900 trials of simulation) during BOLFI, and the entire inference process per individual user took only 1.5 h. This is a tremendous reduction in the computational

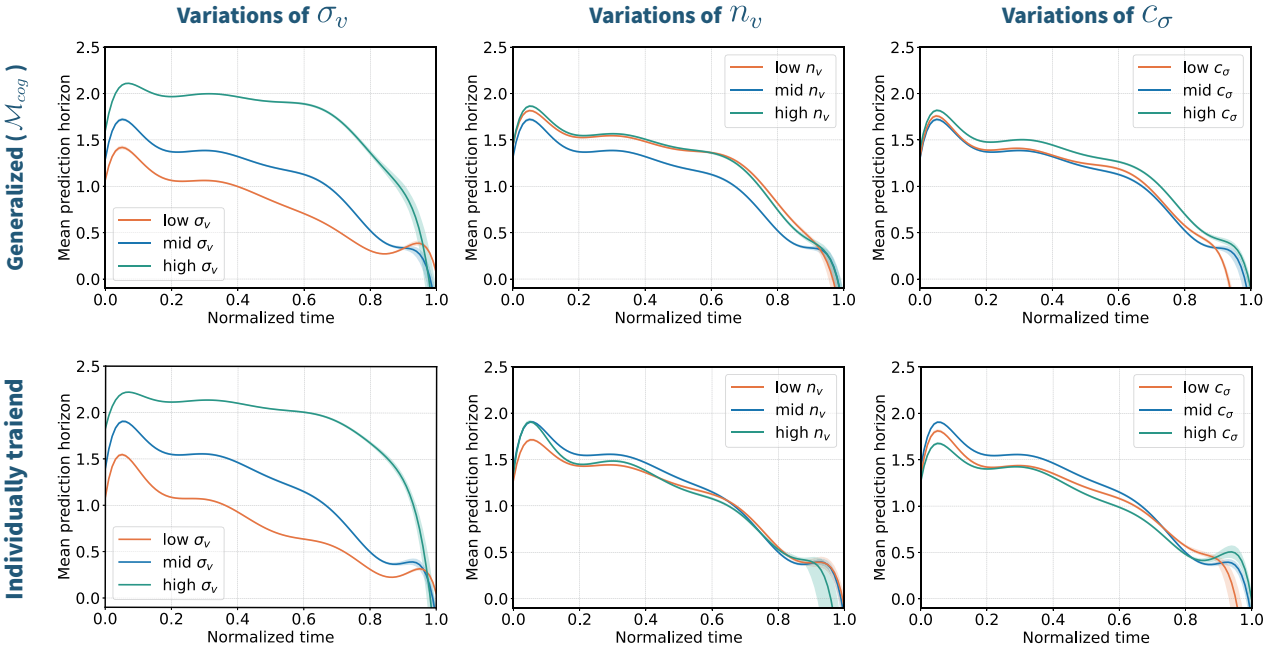

Figure 16: Mean prediction horizon over normalized time of $\mathcal{M}_{cog}$ (*top*) and individually trained models (*bottom*). We investigated the changes in prediction horizon with variations in the value of each cognitive parameter (*orange*: low, *blue*: middle, *green*: high). Each column represents the varying cognitive parameter (*left*: σ_v, *center*: n_v, *right*: c_σ). Each line (mean prediction horizon over time) was obtained by averaging the results from simulations of 12,000 trials of the Accuracy task instruction.

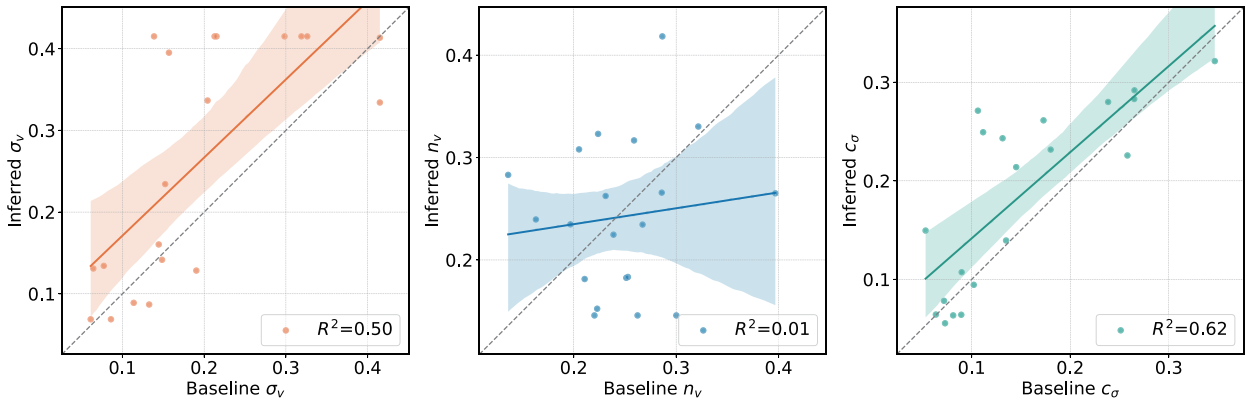

Figure 17: Baseline (x-axis) and our inferred (y-axis) values of each participant's cognitive parameters (*left*: σ_v, *center*: n_v, *right*: c_σ). Linear regression results are presented as solid lines along with the bands of 95% confidence interval.

cost, compared to the hundreds of hours required for the single inference in a previous study [34].

8.4.4 Inference range of cognitive parameters. The appropriateness of the set range can influence inference performance. The range should be sufficiently wide to contain potential candidates. However, if the range is set excessively wide, the parameter space to be searched through BOLFI increases (the number of samples required for inference increases), and the probability of inferring incorrect values increases because it is easy to encounter the local minimum problem during optimization. We defined the range of each cognitive parameter for inference based on baseline values collected from participants. In realistic inference scenarios where the baseline values cannot be acquired, the value ranges that have been empirically validated in previous studies can be used.

8.4.5 Potential degrading factors for inference. Several potential factors can degrade inference performance. First, we assumed that all participants had the same reward formulation for each task instruction obtained in Study 2. However, each individual may have a

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.