A Simulation Model of Intermittently Controlled Point-and-Click Behaviour

Seungwon Do
KAIST, Korea
seungwon.do1@gmail.com

Minsuk Chang[*]
KAIST, Korea
minsuk@kaist.ac.kr

Byungjoo Lee[†]
Yonsei University, Korea
byungjoo.lee@yonsei.ac.kr

ABSTRACT

We present a novel simulation model of *point-and-click* behaviour that is applicable both when a target is stationary or moving. To enable more realistic simulation than existing models, the model proposed in this study takes into account key features of the user and the external environment, such as intermittent motor control, click decision-making, visual perception, upper limb kinematics and the effect of input device. The simulated user's point-and-click behaviour is formulated as a Markov decision process (MDP), and the user's policy of action is optimised through deep reinforcement learning. As a result, our model successfully and accurately reproduced the trial completion time, distribution of click endpoints, and cursor trajectories of real users. Through an ablation study, we showed how the simulation results change when the model's sub-modules are individually removed. The implemented model and dataset are publicly available.

CCS CONCEPTS

• **Human-centered computing → HCI theory, concepts and models**.

KEYWORDS

point-and-click, simulation model, user performance, Fitts' law

ACM Reference Format:
Seungwon Do, Minsuk Chang, and Byungjoo Lee. 2021. A Simulation Model of Intermittently Controlled Point-and-Click Behaviour. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3411764.3445514

1 INTRODUCTION

In human–computer interaction (HCI), quantitative models of user performance can predict how the user performance changes when the interface design or the user characteristics change. Based on the model's predictions, we can mathematically and computationally explore the design space of an interface to obtain the optimal design that maximises user performance, which can drastically reduce the time and cost of developing user interfaces. For example, Fitts' law [58]can predict the time it takes for a user to acquire a target with respect to its size and distance given to the user. From this, in examples such as graphical user interfaces [6, 50], web pages [61, 62] or typing interfaces [34, 52, 70], layouts in which targets are placed on the screen can be optimised or personalised without trial and error.

Existing user performance models in HCI have mainly aimed at predicting aggregated metrics related to user performance, such as trial completion time [2, 8, 58] and error rate [9, 27, 43, 67]. However, such models do not predict how interactions will unfold over time. In response to the limitations of existing models, interest in simulation models [16, 31, 32, 48, 51] has recently increased among HCI researchers. Unlike traditional models, simulation models aim to predict and simulate the progress of interactions in continuous time [1]. This is achieved by modelling the cognitive or physical *entities* that make up the interaction system, each entity's *activities* and the *organisation* between them, as close to reality as possible [18]. For example, to build a simulation model of the user's button press behaviour, Oulasvirta et al. [51] combined individual models such as the user's finger flesh, bones, muscles, button springs and dampers, and the user's perceptual control and Bayesian learning process into a single dynamic system. From this, they successfully simulated the process of a user pressing a button and identified the unknown effect of button design on user performance.

This study presents a novel simulation model that can predict the behaviour and performance of users performing *point-and-click* tasks. In a point-and-click task, the cursor moves to a target using an input device such as a computer mouse and then finally clicks the button on the input device to acquire a target. Point-and-click tasks are still one of the most important and widely given HCI tasks in the modern desktop environment [17]. However, there has not yet been an integrated and realistic point-and-click simulation model. In particular, existing models [48] do not take into account the key characteristics of human motor control (i.e., predictive intermittent control) or did not consider the cognitive process by which users plan and execute click actions.

The user simulated in our model is implemented as a combination of five sub-modules (see Figure 1). The *visual perception module* takes sensory information about the cursor and target as input, adds the user's perceptual noise, and passes the result to the motor control module. The *motor control module* implements the user's intermittent motor control process with a receding horizon that outputs the cursor's motor plan based on the input from the visual

[*]Current affiliation: Naver AI Lab, Korea
[†]Corresponding author

CHI '21, May 8–13, 2021, Yokohama, Japan
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445514

[1]Although not covered in detail in this paper, Model Human Processor [13], GOMS [30], and ACT-R [3] were early symbolic models that tried to predict interaction processes in discontinuous time. Note that this study was inspired by such pioneering studies.

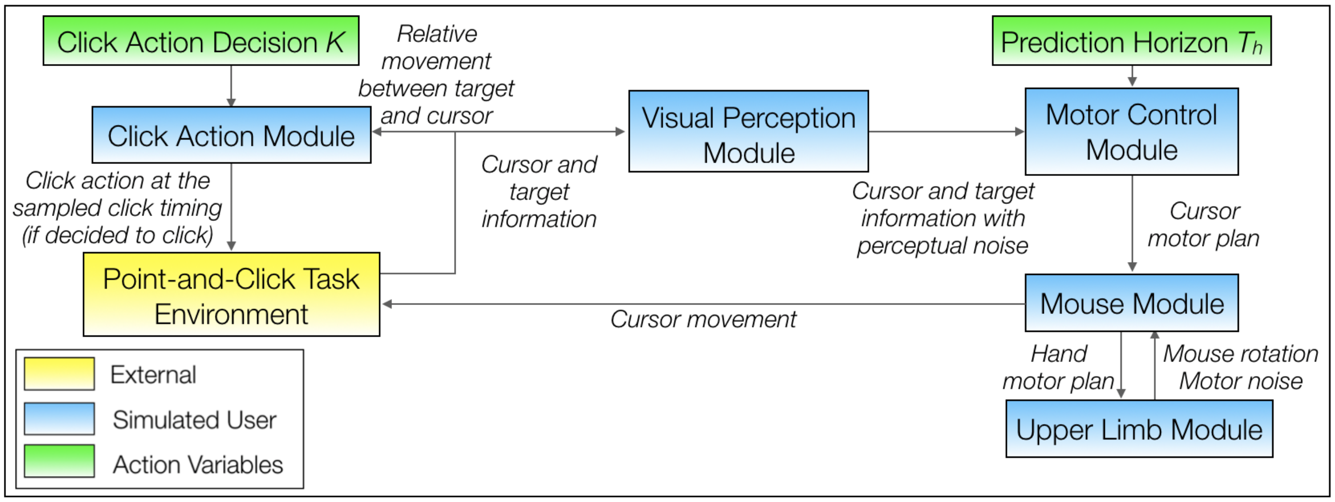
Click Action Decision K
Relative movement between target and cursor
Prediction Horizon T_h
Click Action Module
Visual Perception Module
Motor Control Module
Click action at the sampled click timing (if decided to click)
Cursor and target information
Cursor and target information with perceptual noise
Cursor motor plan
Point-and-Click Task Environment
Cursor movement
Mouse Module
Hand motor plan
Mouse rotation Motor noise
Upper Limb Module
External
Simulated User
Action Variables

Figure 1: In this study, we present a realistic simulation model of point-and-click behaviour. The simulated user is implemented as a combination of five sub-modules, each of which is implemented based on existing models of human visual perception, motor control, motor performance, click action, and input device effects. The simulated user can dynamically control the following two action variables in response to a given point-and-click situation: click decision K and prediction horizon of motor control T_h. The policy of the action is optimised through deep reinforcement learning.

perception module. The process of converting the cursor's motor plan into actual cursor movement is performed by the *mouse module* and the *upper limb module*. In this process, the acceleration function of the mouse, rotation of the mouse coordinate system, and motor noise of the hand movement are considered. The *click action module* continuously observes the relative movement between the cursor and the target and executes a click action at an appropriate timing. Each module is implemented based on existing models of human visual perception [60], motor control [11], motor performance [44, 56], click action [53], and input device effects [14, 40]. The model has 12 free parameters to represent the user's physical, cognitive, and motor characteristics (Table 1). Section 3.2 contains a more detailed description of each module.

The process by which the simulated user performs a point-and-click action in a given environment can be expressed as a Markov decision process (MDP), a natural framework for formulating sequential decision-making processes. For each j-th decision step, there are two variables that represent the simulated user's action $\mathbf{a}_j$ (prediction horizon T_h and click decision K), and six variables that represent the environmental states $\mathbf{s}_j$ that the user perceives (position and velocity of cursor and target, radius of target and hand position). As a result of the action at each decision step, the user receives a reward based on the amount of muscle effort and the success or failure of the click action. We trained the simulated user's policy of action $\pi(\mathbf{s}_j, \mathbf{a}_j)$ to maximise the expected rewards through deep reinforcement learning (a.k.a., deep Q-learning) [47]. Unlike traditional point-and-click studies in which the target was assumed stationary, we trained the simulated user in a more general point-and-click task that included both when the target was stationary and when the target was moving at a constant velocity.

In the evaluation study, the trained model accurately reproduced real user behaviour and performance in terms of trial completion time, distribution of click endpoints, and cursor trajectory. In the ablation study, we analysed how the simulation performance changes when sub-modules of a model are individually removed. In addition, by conducting a subjective evaluation study, we found that the cursor trajectory that the model simulated was difficult to distinguish from that of a real user (mean success rate 57.9 %). To facilitate future research, we released all models and datasets [2].

2 RELATED WORK

2.1 Simulating Target Tracking Movement

User point-and-click behaviour can be divided into a *tracking process* in which the user moves a cursor to a target and a *click process* in which the user performs a click action. Among them, the tracking movement can be simulated in continuous time through *control theoretic models*. Muller et al. [48] simulated the process of a user moving the cursor toward a fixed target assuming that the user is a traditional controller such as a second order lag or McRuer's pilot model [45]. Bachynskyi et al. [5] conducted a similar study for mid-air pointing movements. Aranovskiy et al. [4] proposed a switched dynamic model that could simulate the ballistic phase and tracking phase of point-and-click movement. They formulated the user's visual perception as a linear filter, unlike previous models. Fischer et al. [24] simulated cursor trajectories by assuming that the user is an optimal controller and modelling the cursor as a simple spring-damper system. These models simplified users as controllers with aggregated characteristics in a top-down manner, rather than dissecting the underlying mechanism of point-and-click behaviour.

Unlike the models above, the Basic Unit of Motor Production (BUMP) model suggested by Bye et al. [10–12] is a control theoretic model that incorporates low-level cognitive processes in which the

[2]http://leebyungjoo.com/point-and-click-simulator

user performs the tracking movement. This model explains human-aimed movement as a series of BUMPs; the model considers the following processes in the simulation: intermittent and predictive motor control, signal-dependent motor noise, optimal motor planning, and visual perception of the position and velocity of the target and cursor. The BUMP model both faithfully reproduces human-aimed movements and facilitates integration with other models due to the low-level mechanism it reveals. These strengths mean that the model proposed in this study utilises the BUMP model as a core component. Section 3.2 describes the BUMP model in more detail.

2.2 Simulating Click Action

In point-and-click tasks, users must perform a click action to finally acquire the target. Since click actions are simple movements performed in a short time, they have not received much attention from researchers compared to efforts to explain tracking movement [53]. For example, the effect of the click action has been regarded as simply adding a constant time to the movement time [35] or has been regarded as a non-information component that does not affect the user's channel capacity [69].

The intermittent click planning (ICP) model proposed by Park and Lee [53] revealed that users must go through a significant decision-making process in the process of planning and executing a click action. According to the model, the user perceives the relative movement between the cursor and the target to estimate the optimal timing to activate the click is. Due to the user's intermittent control process, the user's click timing estimation is based on sensory signals obtained during execution of the latest motor plan just prior to the click. This model accurately predicts the distribution of click endpoints and click failure rates. Since the model takes into account the user's intermittent control process, it can be easily combined with the BUMP model. The model proposed in this study utilises the ICP model as another core component along with the BUMP model. Section 3.2 describes the ICP model in more detail.

2.3 Optimising Point-and-Click Policy of Action

Combining the target tracking model and the click action model does not immediately reproduce realistic point-and-click behaviour, simply because human behaviour is not determined in a vacuum but fluidly adapts to the situation of the external environment. Here, the high-dimensional function that determines the myriad mapping between the given states of the task environment and the actions taken by the user is called the user's *policy of action*.

To determine the simulated user's policy of action, it is convenient to formulate the point-and-click process as a Markov decision process (MDP). Under the MDP formulation, the simulated user perceives the external environment, performs actions, and receives rewards for those actions. Then, through deep reinforcement learning [47], the user's policy of action can be optimised to maximise the expected reward. Such optimisation is essential for a realistic simulation of user behaviour. In fact, many existing theories such as Bayesian decision theory [38], cue integration theory [21], and theory of optimal feedback control [63], are based on assumptions about human optimality and are known to be effective at explaining actual human behaviour.

Reinforcement learning has been applied in studies of robotics [20, 22] and human-aimed movements [28, 33]. However, research on the application of reinforcement learning in the context of HCI has only recently started. Cheema et al. [16] simulated mid-air pointing movement by building a dynamic model of the upper limb. Then, after formulating the movement effort as a reward, they optimised the action policy that controlled the joint torque through deep reinforcement learning. They reported an interesting finding that a more realistic simulation is possible when user fatigue is considered in the reward. However, their model does not take into account the user's intermittent motor control process or click action process, so it has a different scope of simulation to ours.

Inspired by such previous studies, the model proposed in this study also formulates the process of the simulated user performing point-and-click as a MDP and obtains the optimal action policy through deep reinforcement learning.

2.4 Other Factors Relevant to Point-and-Click Performance

Besides target tracking and click mechanisms, there are additional factors to consider for a more realistic point-and-click simulation, including human eye movement [55], foveal vision [25], visual perception [60], upper limb kinematics [40], target recognition [49], choice reaction [39], mouse rotation [40], mouse acceleration function [14, 15, 42] and mouse sensor position [36]. Among them, human visual perception, upper limb kinematics, mouse rotation, and mouse acceleration are considered in the model proposed in this study.

3 THE POINT-AND-CLICK SIMULATION MODEL

The proposed simulation model should be understood in three aspects: (1) a simulated user capable of performing point-and-click, (2) the task environment given to the simulated user, (3) the policy of action of the simulated user. In this section we first clearly define the point-and-click task we want to simulate and then move on to a detailed description of the simulated user.

3.1 Task Formulation

3.1.1 Environment and Ergonomics. We assume that a right-handed adult with an average limb length performs a point-and-click task in a typical desktop environment (see Figure 2). The simulated user looks straight at the centre of the monitor and the distance to the monitor is assumed to be 63cm [29]. The height of the user's elbow is equal to the height of the desk surface on which the mouse is placed. It was assumed that the friction on the desk surface is negligible. The simulated user is assumed to operate a standard computer mouse that is light enough to have no difficulty in moving. The mouse is superimposed longitudinally on the user's hand in a neutral position and does not slide or rotate in the user's hand while moving. The mouse sensor is located below the centre point of the user's hand and the acceleration function [14] of the mouse is assumed to be the default slider setting provided by `Mac OS X` 10.12. The size of the monitor is 46.08 cm × 25.92 cm and has sufficient resolution to display the cursor and target.

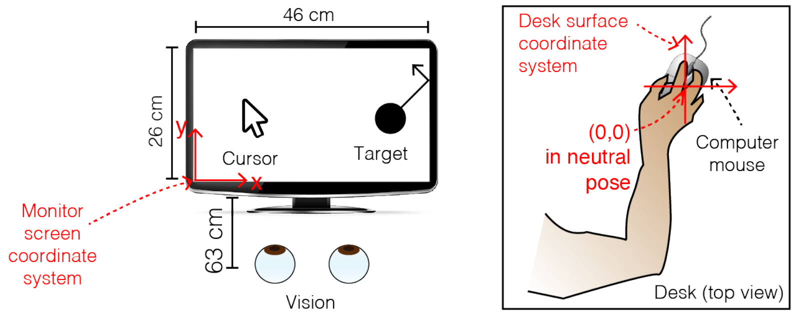

Figure 2: The environment of the point-and-click task given to the simulated user in this study

3.1.2 Task. The simulated user has to move the mouse to place the cursor inside a circular target on the screen and then generate a click event. The target can either move at a constant velocity or be fixed. If the target moves and hits the edge of the screen, it bounces as though reflected and maintains its speed. If the user raises a mouse button click event when the user's cursor is positioned within the target, the trial is considered successful. Otherwise, it is considered to have failed. Immediately after the user activates a click, a new target with random velocity (speed in the range 0–0.509 m/s) is created with a random size (radius in the range 9–24 mm) at a random location on the screen regardless of whether the click was successful or not. The background colour of the task and the colour of the target and cursor have sufficiently high contrast values so that the user can easily distinguish them (e.g., white target on a black background).

3.2 The Simulated User

The simulated user consists of five modules: (1) motor control module, (2) click action module, (3) visual perception module, (4) upper limb module, and (5) mouse module. Each module has the following functions:

- The `motor control module` creates a cursor motor plan ($\mathcal{C}$) that can move the cursor to the target.
- The `click action module` determines whether or not to activate the click and, if so, its timing.
- The `visual perception module` receives the velocity and position of the target and the cursor, and the position of the hand from the environment, and outputs the perceived values including perceptual noise.
- The `mouse module` simulates the process of converting the motor plan of the cursor ($\mathcal{C}$) into the motor plan of the hand ($\mathcal{H}$) and how the cursor will move when the motor plan of the hand is executed.
- The `upper limb module` simulates the rotation of the mouse by assuming the arm is a two-dimensional (2D) three-degree-of-freedom (3-DOF) manipulator with three revolute joints (shoulder, elbow, and wrist). This module also simulates the Gaussian additive motor noise proportional to the hand movement speed.

This section details the actual implementation of a simulated user. First, we describe the simulation environment and then each module.

3.2.1 Simulation Environment. The user simulation is performed in a discrete time with constant step ($\Delta t = 0.05$ s). This minimum time unit of the simulation reflects that the EMG burst observed in the human motor control process usually lasts for 50 ms [11]. If the time taken from the start of the simulation ($t = 0$) to the i-th time step is t_i, the position and velocity of the cursor ($\mathbf{p}_c$ and $\mathbf{v}_c$) and the position and velocity of the target ($\mathbf{p}_t$ and $\mathbf{v}_t$) in the i-th time step are represented as follows:

$$\begin{aligned} \mathbf{p}_c[t_i] &= (p_c^x[t_i]\,,\, p_c^y[t_i]) & \mathbf{v}_c[t_i] &= (v_c^x[t_i]\,,\, v_c^y[t_i]) \\ \mathbf{p}_t[t_i] &= (p_t^x[t_i]\,,\, p_t^y[t_i]) & \mathbf{v}_t[t_i] &= (v_t^x[t_i]\,,\, v_t^y[t_i]) \end{aligned} \quad (1)$$

The coordinates of the cursor and target are expressed in meter units based on a coordinate system where the origin is located at the bottom left of the monitor screen (see Figure 2).

The position and velocity vector of the centre of the hand ($\mathbf{p}_h$ and $\mathbf{v}_h$) holding the mouse are expressed as follows:

$$\mathbf{p}_h[t_i] = (p_h^x[t_i]\,,\, p_h^y[t_i]) \qquad \mathbf{v}_h[t_i] = (v_h^x[t_i]\,,\, v_h^y[t_i]) \quad (2)$$

The coordinates of the hand are expressed in meter units based on the desk coordinate system where the position of the hand is (0,0) when the limb is in its neutral position (see Figure 2). The target radius is expressed as R_t in meter units.

3.2.2 Motor Control Module. The motor control module implements an existing model of human intermittent motor control, the BUMP model [10–12], which assumes that the process of planning and executing human-aimed movement consists of a series of BUMPs. A BUMP is further divided into three successive processes: sensory analysis (SA), response planning (RP), and response execution (RE). These three processes do not overlap each other and each take the same time interval T_p (= 0.1 s). BUMPs that are adjacent to each other overlap by T_p and are performed in parallel (Figure 3).

In SA, the simulated user perceives the position and velocity of the target and cursor from a given stream of sensory signals. With perceived information, the agent in RP builds a motor plan to bring the cursor to the target. The motor plan is built only with the information already perceived in the previous SA, and once the RP has started, the newly perceived environmental information is no longer reflected in the current motor plan due to the psychological refractory period (PRP) of the central nervous system (CNS) [57]. Finally, in RE, the agent executes the motor plan obtained from RP.

Of the three processes in BUMP, RP is central. SA and RE are described in more detail along with other modules, whereas here

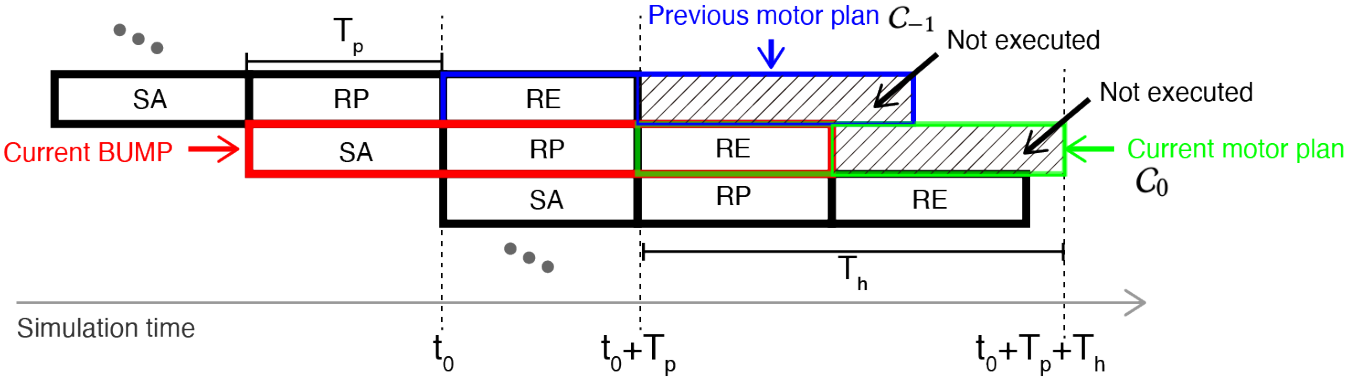

Figure 3: In the BUMP model, the human intermittent motor control process is performed as a series of BUMPs. A BUMP consists of three processes: sensory analysis (SA), response planning (RP), and response execution (RE). Successive BUMPs partially overlap each other. t_0 is the moment when the RP of the current BUMP starts and T_h is the prediction horizon for planning the motor plan in the current RP.

we focus on describing the RP process. Let t_0 be the point at which the *current* RP starts. The RP constructs a motor plan of the cursor to be executed in the subsequent RE based on the two pieces of information perceived from the previous SA. The first information is the position and velocity of the cursor at the end of the current RP. If we put a hat ($\widehat{\ }$) on the perceived value, each of this information can be written in this form: $\widehat{\mathbf{p}}_c[t_0 + T_p]$ and $\widehat{\mathbf{v}}_c[t_0 + T_p]$. The second information is the position and velocity of the target that the simulated user aims to reach with the cursor. The user uses the perception of the target's future position and velocity at the time T_h passes after the current RP is over. Each of this information can be written like this: $\widehat{\mathbf{p}}_t[t_0 + T_p + T_h]$ and $\widehat{\mathbf{v}}_t[t_0 + T_p + T_h]$. Here, T_h is a variable called the *prediction horizon* that represents the degree of predictive control.

The final goal of RP is to create a cursor motor plan C that can make the position and velocity of the cursor and the target the same at the end of the prediction horizon ($t = t_0 + T_p + T_h$). Here, the assumption was introduced that the user will pursue the equality of velocity because it reduces the relative speed between the cursor and the target, thus making it easier for the simulated user to plan and execute the click action. Due to the redundancy in the human motor system [63], there can be a myriad of motor programs that satisfy the condition. The BUMP model resolves this redundancy by assuming optimality of the user; the simulated user is assumed to build an optimal cursor motor plan C that minimises the *mean squared acceleration* of cursor movement:

$$\begin{gathered} C_0 = \mathbf{OTG}(\widehat{\mathbf{p}}_c, \widehat{\mathbf{v}}_c, \widehat{\mathbf{p}}_t, \widehat{\mathbf{v}}_t) = \{\mathbf{p}_c[t_i], \quad \mathbf{v}_c[t_i]\} \\ \text{where, } t_i = (t_0 + T_p) \ \text{ to } \ (t_0 + T_p + T_h) \end{gathered} \tag{3}$$

Here, the subscript 0 indicates that the motor plan C was made from the *current* BUMP's RP. Similarly, the motor plan made from the previous BUMP can be represented as C_{-1}, and the motor plan to be made at the next BUMP can be used as C_{+1}. **OTG** is a function that receives perceptions from SA and returns a cursor motor plan, which represents the *optimal trajectory generation.* **OTG** consists of simple matrix operations; please refer to the original paper for detailed function implementation [10–12]. The Figure 4 below shows the motor plan of the cursor made from **OTG** when the cursor is stopped initially at (0,0) and the future velocity and position of the target are (-0.05, 0) and (0.1, 0.2) (units: m/s and m, respectively).

As soon as RP is finishes, the built motor plan C_0 starts to be executed in the subsequent RE. While C_0 is running, the next BUMP plans a new motor plan C_{+1}, and when it is ready, the agent replaces the existing motor plan with it (i.e., intermittent motor control). For each new BUMP, the prediction horizon T_h must be set. In this study, the simulated user adjusts the T_h value based on the policy of action optimised through reinforcement learning.

3.2.3 *Click Action Module.*

The click action module simulates the process by which the simulated user plans and executes the click action while the cursor is moving toward the target. The module is implemented based on the intermittent click planning (ICP) model [53], the latest model of user click performance. The main tenet of the model is that the process of determining the click action is performed in the middle of the target-tracking movement. In particular, the model assumes that whenever the user's cursor motor plan is intermittently updated, the user establishes a new click plan. This can be explained in conjunction with the BUMP model as

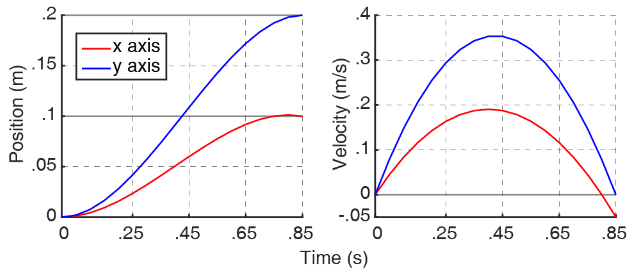

Figure 4: In the RP stage of the BUMP model, the user builds an optimal motor plan that can move the cursor to the target. This process can be implemented through the OTG function, and the resultant cursor trajectory minimises the mean squared acceleration.

follows: Based on the sensory signals given during the execution of the latest motor plan C_0, the user must decide *when* to execute the click action. The model assumes that this decision-making occurs in parallel with the motor control and is performed by the user's *internal clock*.

Let W_t be the expected duration it takes for the cursor to pass through the target during the execution of the current motor plan C_0. The ICP model assumes that the user perceives W_t from the average velocity of the relative motion between the cursor and the target (see Figure 5). Note that the perception of W_t is not based on the ideal motor plan, but rather on the actually executed motor plan, including motor noise and effects of the input device. In this situation, the user's goal is to generate a click input within W_t from the moment the cursor makes first contact with the target.

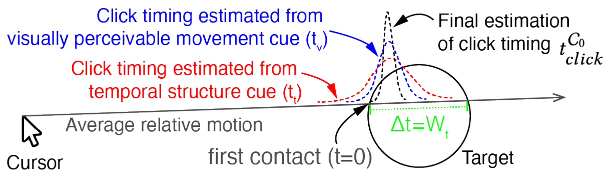
Click timing estimated from
visually perceivable movement cue (t_v)
Final estimation
of click timing $t^{C_0}_{click}$
Click timing estimated from
temporal structure cue (t_t)
Average relative motion
Δt=W_t
Cursor
first contact (t=0)
Target

Figure 5: According to the ICP model, the user integrates two sensory cues given during the execution of the current motor plan in order to estimate the optimal timing of the click action.

According to the ICP model, two sensory cues can be given for the user's internal clock to estimate the optimal click timing. First, by perceiving the relative motion between the target and the cursor, the users can predict when the cursor will contact the target and estimate its click timing; this is called the `visually perceivable movement cue` [41]. Second, if the click action has been repeatedly executed before, the user can estimate when the next click should be from the time period of the repetition. This is called the `temporal structure cue` [41, 43]. Let the click timing estimated from each cue be t_v (from visually perceivable movement cues) and t_t (from temporal structure cues). According to the model, the posterior probability distribution of the user's input timing estimation is generally Gaussian. If the cursor first contacts the target at $t = 0$, then the mean (M) of each estimate can be expressed as follows:

$$\mathrm{M}[t_v] = \mathrm{M}[t_t] = c_\mu \cdot W_t \tag{4}$$

Above, c_μ is a constant and represents the user's implicit aim point within W_t. The model then expresses the standard deviation (STD) of each estimate as follows:

$$\mathrm{STD}[t_v] = c_\sigma(1/(e^{\nu \cdot T_c} - 1) + \delta) \quad \text{and} \quad \mathrm{STD}[t_t] = c_\sigma \cdot P \tag{5}$$

Here, the STD of the t_v increases as T_c shortens. The T_c is called the cue viewing time and is the time from the start of the motor plan's execution to the time the cursor makes contact with the target. The shorter the T_c, the less time the user will have to encode the visual cues, so the estimate will be less reliable (i.e., higher STD). Second, the STD of t_t increases as the time period P in which the click was repeated increases. This is due to the *scalar property* of the human internal clock [26] in encoding the time period of repeated events. This can be seen from common experience; it is much harder to clap once every five seconds than to clap once per second. The c_σ, ν, and δ are constants that represent the cognitive characteristics of the simulated user.

According to *cue integration theory* [65], the user integrates estimates from different cues in a statistically optimal way. If the two sensory cues are assumed to be statistically independent of each other, according to the theory, the mean and standard deviation of the integrated click timing $t^{C_0}_{click}$ obtained for the motor plan C_0 can be expressed as follows:

$$\begin{aligned}\mathrm{M}[t^{C_0}_{click}] &= c_\mu \cdot W_t \\ \mathrm{STD}[t^{C_0}_{click}] &= c_\sigma \cdot P/\sqrt{1 + (P/(1/(e^{\nu \cdot t_c} - 1) + \delta))^2}\end{aligned} \tag{6}$$

Finally, the click action module samples the actual click timing $t^{C_0}_{click}$ from the Gaussian distribution having the above mean and standard deviation while the motor plan C_0 is running.

The shorter W_t or the higher the STD of the click timing estimation, the more difficult it is to succeed on the click. Therefore, the degree of *difficulty of the user's click planning* (D_{click}) can be quantified as follows:

$$D_{click} = \log_2\left(\frac{P/\sqrt{1 + (P/(1/(e^{\nu \cdot t_c} - 1) + \delta))^2}}{W_t}\right) \ \text{(unit: bits)} \tag{7}$$

According to the ICP model, the user's click failure rate increases in proportion with the value of D_{click}. For example, if D_{click} is 1 bits, the user's click failure rate is predicted to be approximately 5 %; for 3 bits, it will be approximately 40 % [43].

Meanwhile, the planning and sampling of the click action described thus far is only performed only when the simulated user *decides to click*. The user's intention to click is represented by a binary variable K. If the module decides not to click ($K = 0$), click planning and sampling do not occur and the next BUMP is performed as usual. If the module decides to click ($K = 1$), the above-described click planning and sampling process described above is performed and there will be no subsequent BUMP. Instead, the current motor plan C_0 continues to run and the module executes an impulsive click action (i.e., zero duration) in the sampled timing $t^{C_0}_{click}$. If the click timing is later than the end of the motor plan, the click is executed immediately after the execution of the motor plan finishes. If W_t is zero because the cursor has not penetrated the target, the click action is executed immediately at the end of the motor plan. The instant the click action is executed, a new BUMP begins for the newly given target.

3.2.4 Visual Perception Module.

For the motor control module to build a cursor motor plan C_0, it needs to estimate the future position and velocity of the cursor and target (see Equation 3). Let t_0 be the moment when the current RP starts. RP must perceive the future position and velocity of the cursor at the moment $t = (t_0 + T_p)$. In addition, RP must perceive the future position and velocity of the target at the moment $t = (t_0 + T_p + T_h)$. The visual perception module implements the estimation process and the perceptual noise added to it. How each perception is implemented is described below.

(1) Perceiving cursor position and velocity: We assume that through the SA process just before the RP, the simulated user can perceive the position of the cursor with sufficiently high precision at the moment the RP is started ($\therefore \widehat{\mathbf{p}}_c[t_0] = \mathbf{p}_c[t_0]$). In addition, the

simulated user knows the ideal motor plan C_{-1}, which was planned in the previous RP and is running in the current RP (i.e., efferent copy). From these, the user can perceive the position and velocity of the cursor at the moment ($t = t_0 + T_p$) as follows:

$$\begin{aligned}\widehat{\mathbf{p}}_c[t_0 + T_p] &= \mathbf{p}_c[t_0] + (\mathbf{p}_c^{C_{-1}}[t_0 + T_p] - \mathbf{p}_c^{C_{-1}}[t_0]) \\ \widehat{\mathbf{v}}_c[t_0 + T_p] &= \mathbf{v}_c^{C_{-1}}[t_0 + T_p]\end{aligned} \tag{8}$$

Here, $\mathbf{p}_c^{C_{-1}}$ and $\mathbf{v}_c^{C_{-1}}$ are the cursor position and velocity retrieved from the previous motor plan C_{-1}. In the above equation, the user perceives the future cursor position by predicting that the displacement when the previous motor plan is ideally executed will be added to the position of the cursor perceived at the moment t_0. For the future velocity of the cursor, the user perceives it as the velocity when the previous motor plan is executed ideally.

(2) Perceiving target position and velocity: We assume that through the SA process just before the RP, the simulated user can perceive the position of the target with sufficiently high precision at the moment the RP starts ($\therefore\ \widehat{\mathbf{p}}_t[t_0] = \mathbf{p}_t[t_0]$). In addition, we assume that the user perceives the target's velocity at the same moment ($\widehat{\mathbf{v}}_t[t_0]$), but there is significant perceptual noise. To simulate such perceptual noise, our model implements Stocker's model of speed perception [60, 66]:

$$p(\ \widehat{s}_t \mid s_t'\ ) = \frac{1}{\widehat{s}_t \sqrt{2\pi}\sigma_s} \exp\left[\frac{-(\ln \widehat{s}_t - \ln s_t')^2}{2\sigma_s^2}\right] \tag{9}$$

In the above equation, s_t' is the normalised target speed. If s_t (= $\|\mathbf{v}_t\|$) is the actual target speed expressed in visual angle (deg/s), s_t' is calculated as follows: $s_t' = 1 + s_t/s_0$. Here, s_0 is a normalisation constant that can be set to 0.3 deg/s according to the original paper [60]. The $\widehat{s}_t$ is the user's estimation of the target speed and $p(\widehat{s}_t|s_t)$ is the likelihood of the target speed estimated by the user, as shown in the above equation, which follows a log-normal distribution. Here, σ_s represents the width of the distribution, and the larger it is, the less reliably the user can estimate the target's speed. According to the model, σ_s decreases as the contrast between the target and the background increases, and since sufficient contrast was assumed in this study, the value was set to 0.15 by referring to the original paper [60].

Eventually, the target velocity perceived at t_0 by the user can be written as:

$$\widehat{\mathbf{v}}_t[t_0] = (\widehat{s}_t[t_0] - 1) \cdot s_0 \cdot \mathbf{v}_t[t_0]\ /\ \|\mathbf{v}_t[t_0]\| \quad \text{(unit: deg/s)} \tag{10}$$

Here, it is assumed that noise is only included in the perception of magnitude rather than direction. In addition, note that we converted the normalised speed back to the original unit (deg/s). This can be converted into m/s later for simulation by considering the distance between the screen and the user (0.63 m).

Finally, the module assumes that the simulated user *extrapolates* the target's future position based on the perception of the target velocity at t_0 as follows:

$$\widehat{\mathbf{p}}_t[t_0 + T_p + T_h] = \mathbf{p}_t[t_0] + (T_p + T_h) \cdot \widehat{\mathbf{v}}_t[t_0] \tag{11}$$

For the target's future velocity, it is assumed that the user expects that the target will continue maintaining their velocity perceived at t_0:

$$\widehat{\mathbf{v}}_t[t_0 + T_p + T_h] = \widehat{\mathbf{v}}_t[t_0] \tag{12}$$

We assume that the simulated user can estimate the future position and velocity of the target including the effect that the target bounces at the edge of the screen without any additional perceptual noise.

3.2.5 *Mouse Module.*

The motor plan generated from the motor control module governs the *cursor* movement. However, the user must move the mouse with their hand to control the cursor, so it is necessary to explain the process of how the user converts the motor plan of the cursor C_0 into the motor plan of the hand $\mathcal{H}_0$. The mouse module implements two components to simulate this: (1) mouse acceleration function [14] and (2) mouse coordinate disturbance [40].

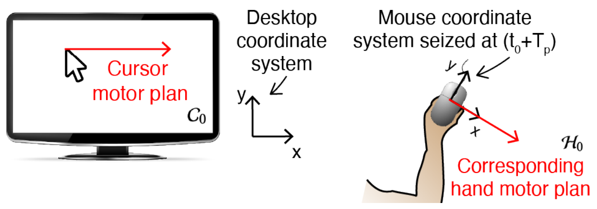

Figure 6: The simulated user has to convert the cursor's motor plan to the hand's motor plan. In this process, the user uses the mouse coordinate system captured at the end of RP as the reference coordinate system for the conversion.

Before explaining the conversion process, we need to specify in which *coordinate system* the user will plan $\mathcal{H}_0$. There are two possibilities; first, the user can plan it based on a fixed coordinate system on the desk. Second, the user can plan it based on the sensor coordinate system fixed to the mouse. According to a previous study [68], users decide the direction of the input based on the field of view they expect to get when looking at the input device (i.e., the visual field compatibility). Therefore, the model assumes that the user will convert C_0 into $\mathcal{H}_0$ based on the mouse coordinate system rather than the desk coordinate system. However, the mouse coordinate system continues to rotate while the hand is moving [40]. We assume that the user perceives how the mouse coordinate system will be rotated at the end of the RP ($t = t_0 + T_p$) and converts C_0 to $\mathcal{H}_0$ with respect to the coordinate system (see Figure 6). It is assumed that this process does not include perceptual noise.

(1) Mouse acceleration function: The mouse acceleration function is a function that maps the speed of the mouse body or the user's

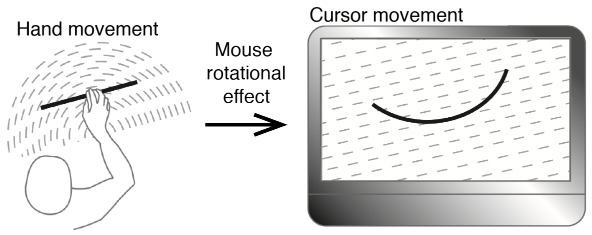

Figure 7: If the mouse rotates while the hand is moving, the cursor movement will be pulled in the y-axis direction more than the user intended; this is called coordinate disturbance and occurs during mouse control.

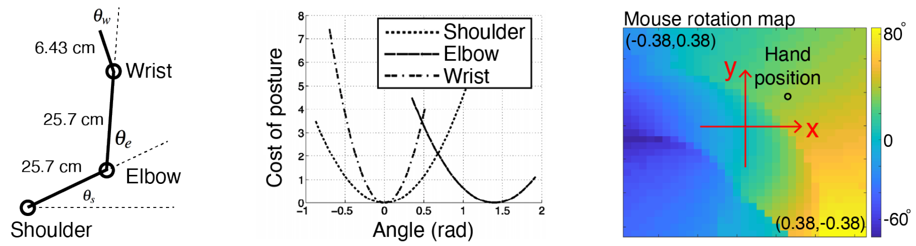

Figure 8: The upper limb of the simulated user was modelled as a 3-DOF manipulator (left). The cost perceived by the user is in accordance with the rotational degree of each joint (middle, copied with permission from [37]). The degree of rotation of the mouse depends on where the hand is placed on the desk (right)

hand ($\|\mathbf{v}_h\|$) to the speed of the cursor ($\|\mathbf{v}_c\|$) [14, 42]. The function (f_{acc}) can generally be expressed as:

$$\|\mathbf{v}_c\| = f_{acc}(\|\mathbf{v}_h\|) \tag{13}$$

The acceleration function is usually set differently depending on the input device, operating system (OS), and OS version [14]. We replicated the simulated user's acceleration function (`Mac OS X` 10.12, system default slider setting) using the `Libpointing` library [14].

The module assumes that the simulated user can determine the hand speed inversely, taking full account of the mouse acceleration function:

$$\begin{aligned} &\text{For each } \mathbf{v}_c[t_i] \in C_0 \\ &\mathcal{H}_0 \leftarrow \mathbf{v}_h[t_i] = f_{acc}^{-1}(\|\mathbf{v}_c[t_i]\|) \cdot \mathbf{v}_c[t_i]/\|\mathbf{v}_c[t_i]\| \end{aligned} \tag{14}$$

From this transformation, the velocity vector of the hand is in the same direction as the velocity of the cursor (w.r.t. the different coordinate system), but has a different magnitude considering the effect of the acceleration function inversely. The hand position plan is obtained by integrating the velocity plan as follows:

$$\mathcal{H}_0 \leftarrow \mathbf{p}_h[t_i] = \mathbf{p}_h[t_0 + T_p] + \sum_{t=t_0+T_p}^{t_i} \mathbf{v}_h[t_i] \tag{15}$$

Note that to simplify the model, it is assumed that the position of the hand at the end of the RP ($\mathbf{p}_h[t_0 + T_p]$) is perceived by the user without noise.

(2) Mouse coordinate disturbance: When the motor plan of the hand obtained from the RP is executed in the subsequent RE, the cursor moves accordingly. The speed of the hand can be converted into the speed of the cursor using the acceleration function (see Equation 13). However, determining the direction of cursor movement is not straightforward. According to a previous study [40], since the mouse rotates while the user moves the mouse, the motor plan of the hand is not converted into the movement of the cursor as the user intended. This is called *coordinate disturbance* in mouse control. When the mouse rotates clockwise (counterclockwise) while the hand is moving, the velocity vector of the cursor rotates slightly counterclockwise (clockwise) every step compared to the hand velocity vector. This leads to a tendency for the cursor trajectory to be pulled in the y-direction compared to the hand trajectory (see Figure 7).

Referring to the previous study [40], the hand motor plan $\mathcal{H}_0$ ($\mathbf{p}_h[t_i]$ and $\mathbf{v}_h[t_i]$) can be transformed into the motion of the cursor, taking into account both the acceleration function and the coordinate disturbance as follows:

$$\begin{aligned} \mathbf{v}_c[t_i] &= \frac{f_{acc}(\|\mathbf{v}_h[t_i]\|)}{\|\mathbf{v}_h[t_i]\|} R\Big[\frac{-\Delta\theta_m}{2} - \theta_m\Big] \mathbf{v}_h[t_i] \\ \mathbf{p}_c[t_i] &= \mathbf{p}_c[t_0 + T_p] + \sum_{t=t_0+T_p}^{t_i} \mathbf{v}_c[t_i] \end{aligned} \tag{16}$$

Here, R is the 2D rotation matrix and θ_m is the expected amount of mouse rotation at a specific position on the desk and is thus a function of hand position $\mathbf{p}_h$. $\Delta\theta_m$ is the amount of change in θ_m between the previous time step (t_{i-1}) and the current time step (t_i). θ_m is the output from the upper limb module, which is described in a later section.

3.2.6 Upper Limb Module. The upper limb module simulates two phenomena: (1) rotation of the mouse that occurs when the simulated user moves their hand and (2) motor noise added to hand movement. Each is described below.

(1) Simulating mouse rotation: In our model, the mouse is attached longitudinally to the simulated user's hand. Therefore, to simulate how much the mouse will rotate, we need to simulate how the human upper limb will move during the mouse control. For this, the module assumes that the user's upper limb is a 2D 3-DOF manipulator (see Figure 8). In the neutral position, the angle of rotation for each joint is: shoulder (θ_s =0°), elbow (θ_e =80°), wrist (θ_w =0°). According to previous studies [19, 40], the subjective discomfort felt by the user for each joint can be expressed as a parabolic function of the rotation angle: $C(\theta_s)$, $C(\theta_e)$ and $C(\theta_w)$. From this, we can solve the inverse kinematics of the limb to minimise the total sum of discomfort. As a result, Figure 8 plots the expected angle of orientation of the mouse (θ_m) for each hand position on the desk surface.

(2) Signal-dependent motor noise: While the hand motor plan $\mathcal{H}$ is running, the upper limb module also implements the signal-dependent motor noise [56]. This noise is implemented as a Gaussian random noise with a standard deviation proportional to the hand speed (i.e., $\sigma = n \cdot \|\mathbf{v}_h\|$). At each time step, noise sampled independently in the parallel and perpendicular directions was added to the hand velocity vector. The proportional constants in parallel ($n_{\|}$) and perpendicular ($n_{\perp}$) directions were set differently and were 0.2 and 0.02, respectively.

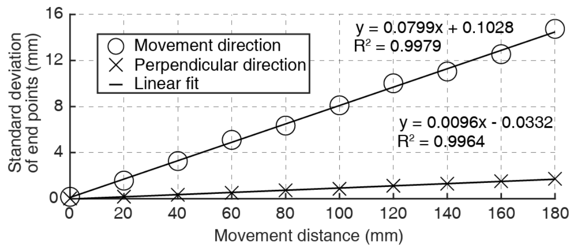

Figure 9: The upper limb module simulates Gaussian motor noise that increases in proportion to the movement speed. This graph shows how the user's motor noise is implemented in both the direction of the hand's movement and the direction perpendicular to it. The simulated user moved to the target at the distance shown on the x-axis in 0.2 seconds. This faithfully reproduces the experimental results of previous research [44].

Lin et al. [44] have studied motor noise of users when performing ballistic aimed movements using a computer mouse. They gave participants in the experiment the short time of 0.2 seconds to move the cursor by aiming at a target that was a certain distance away. Then, at the end of the aimed movement, the distance between the target centre and the cursor end point was reported for the direction parallel to and perpendicular to the movement vector. To compare our motor noise with their results, we set T_h to 0.2 s for the first BUMP and 0.1 s for the second BUMP so that the simulated user could reach the target in 0.2 seconds. From the simulation, we confirmed that the output of our simulation on the same task faithfully reproduced their results (see Figure 9).

3.2.7 Free Parameters of The Simulated User. There are 12 free parameters that determine the simulated user's cognitive and behavioural characteristics (see Table 1). Setting these parameters differently permits the simulation of various users with different cognitive and behavioural characteristics. In this study, we simulate an *average user*; therefore, the free parameters were set to the typical mean values reported in previous studies.

3.3 Optimising Policy of Action

We can formulate the simulated user's point-and-click behaviour as a Markov decision process (MDP). MDP [7] is a mathematical model for formulating an agent's sequential decision-making problem. An MDP describes the sequential decision-making process using three components, *state*, *action*, and *reward*. At each discrete decision-making step, the decision-making agent perceives the *state* of the external environment and performs an *action*, receives a *reward* as a result of the action, and arrives at a new *state*. The goal of this agent is to maximise the cumulative reward during the entire decision horizon, called an *episode*.

In our model, the simulated user's decision-making takes place every j-th BUMP and the set of decision-making series from the time the target is given until the click is made is an episode. At each j-th BUMP, the state $\mathbf{s}_j$ perceived by the simulated user, the user's action $\mathbf{a}_j$ and the reward $r(\mathbf{s}_j, \mathbf{a}_j)$ given to the user are expressed as follows:

- State $\mathbf{s}_j$: perceived future cursor position $\widehat{\mathbf{p}}_c[t_0 + T_p]$ and velocity $\widehat{\mathbf{v}}_c[t_0 + T_p]$, perceived future target position $\widehat{\mathbf{p}}_t[t_0 + T_p + T_h]$ and velocity $\widehat{\mathbf{v}}_t[t_0 + T_p + T_h]$, future hand position $\widehat{\mathbf{p}}_h[t_0 + T_p]$, target radius R_t. These are continuous values.
- Action $\mathbf{a}_j$: prediction horizon T_h (0.1 to 2.5 s with 0.1 s interval) and click decision K (0 or 1)
- `Reward` $r(\mathbf{s}_j, \mathbf{a}_j)$: If the user takes an action $\mathbf{a}_j$ in a state $\mathbf{s}_j$, the user will receive the following reward:

$$r = \begin{cases} -\sum_{t=t_0+T_p}^{t_0+2T_p} \|\dot{\widehat{\mathbf{v}}}_h[t]\| & \text{if } K = 0 \\ R^+ - \sum_{t=t_0+T_p}^{t_0+T_p+T_h} \|\dot{\widehat{\mathbf{v}}}_h[t]\| & \text{if } K = 1\&\text{click success} \\ R^- - \sum_{t=t_0+T_p}^{t_0+T_p+T_h} \|\dot{\widehat{\mathbf{v}}}_h[t]\| & \text{if } K = 1\&\text{click fail} \end{cases}$$

  Here, $\dot{\widehat{\mathbf{v}}}_h$ is the acceleration of the hand in the motor plan $\mathcal{H}_0$ before the signal-dependent motor noise is added. The sum of the absolute acceleration values represents the user's *motor execution effort*. Note that this effort term is only obtained for the time interval in which the motor plan will actually be executed. R is a reward ($R^+ = 14$) or penalty ($R^- = -1$) given according to the success or failure of the click.

Note that the state vector above is defined based on the simulated user's perceptual variables, while it is usually defined as the true state variables of the environment. Nevertheless, it still satisfies the Markov property; that is, the action ($\mathbf{a}_j$) and state ($\mathbf{s}_j$) in the j-th BUMP determine the probability distribution of the state ($\mathbf{s}_{j+1}$) in the $(j+1)$-th BUMP. This allows the problem to be safely formulated as a MDP. In our model, the transition function $T(\mathbf{s}_{j+1}|\mathbf{s}_j, \mathbf{a}_j)$ is determined through the simulation process described in the preceding sections.

The value of an action in a specific state is calculated as a summation of the current and next state rewards. The next reward is less valuable than the current reward by a discount factor γ, which we set to 0.95.

3.4 Reinforcement Learning in MDP

The purpose of the MDP formulation is to obtain the optimal action policy π – a probabilistic rule that determines which action the decision-making agent should perform for a given state – of the simulated user. We can achieve the goal by employing a reinforcement learning algorithm that learns an optimal policy π that earns the *maximum cumulative reward.* To compare different policies - and thus calculate the optimal policy - many reinforcement learning algorithms use the notion of "Q function". Q functions measure the goodness of a particular state and action pair: how good is it for an agent to employ a particular action at a particular state. By definition, it is the expected discounted rewards that the agent collects totally by following the specific policy until the end of the decision horizon.

If the MDP formulation has a large state and action space, visiting every state for each employable policy and executing every action to calculate the Q function becomes quickly infeasible. Function approximation can be useful in such a case. In particular, Deep Q-Network [46] has been proven to be efficient and effective for learning the optimal policy. DQN is known to be sample efficient in discrete action space, which is beneficial for our simulation setup. In addition, DQN has the advantage of being applicable to cases where

Table 1: This table shows the free parameters that describe the simulated user's cognitive and behavioural characteristics. The parameter values were set by referring to previous studies assuming an average user.

Variable	Description	Value	Ref	Module
T_p	Planning time interval	0.1 s	[11]	Motor control
n_v	Motor noise constant (parallel)	0.2	[44]	Upper limb
n_p	Motor noise constant (perpendicular)	0.02	[44]	Upper limb
l_{se}	Shoulder-to-elbow length	25.7 cm	[40]	Upper limb
l_{ew}	Elbow-to-wrist length	25.7 cm	[54]	Upper limb
l_{wh}	Wrist-to-hand length	6.43 cm	[40]	Upper limb
σ_v	Width of likelihood of visual speed perception	0.15	[60]	Visual perception
$f_{gain}()$	Mouse acceleration function	OS X 10.12	[14]	Mouse
c_σ	Precision of internal clock	0.09015	[41, 53]	Click action
c_μ	Implicit aim point	0.185	[41, 53]	Click action
ν	Drift rate	19.931	[41, 53]	Click action
δ	Visual encoding precision limit	0.399	[41, 53]	Click action

the state space is much larger or even the state space is continuous. This allows the proposed simulation model to be extended to more general situations in the future, for example, the visual module receives raw pixels of the screen as input.

A deep Q-Network uses a neural network to approximate the Q functions. Specifically, the neural network in the DQN takes the current state as the input and outputs a vector containing the Q (function) value of every action. Then, DQN uses an epsilon-greedy algorithm to probabilistically choose between the action with the highest Q value and a random action. In simple terms, the DQN algorithm iterates the following steps to learn the optimal policy:

(1) Under the agent's current policy, gather and store samples in a "memory" called a replay buffer. The samples are called "experiences".
(2) Randomly sample batches of experiences from the replay buffer. This part is called the "experience replay".
(3) Use the sampled experiences to update the Q network values, which will later be used to choose the "optimal action", and ultimately describe the optimal policy.
(4) Repeat steps 1–3 until agent's reward converges.

When training the DQN, the "experience replay"in step 2 alleviates the possible correlations in the consecutive training examples in most recent transitions, which otherwise would drive the network into a local minimum. The details of the algorithm, including its pseudocode, can be found in the work by Mnih et al. [46].

3.4.1 Training Implementation and Setup. For the approximation of the Q function, we trained the neural network with an input layer including 11 units which is the number variables that describes our definition of a state (five 2D vectors and one scalar), one fully connected hidden layer (512) and an output layer with 50 units which is the number of available actions at each state (25 $T_h \times 2$ K). The neural network weights were initialised with random seeds. We used Adam [37] for learning the weights of the neural network. The simulated user performed four million episodes of the point-and-click task defined in Section 3.1.2. Setting the hyperparameters, we followed the original DQN experiments [46]. However, differences existed between the two setups, and we adjusted accordingly. For example, the original DQN paper took images data through convolution as the input, whereas ours was a well-defined point-and-click scenario. Our ground rule for fine-tuning was to search for a parameter set with which the loss values in the loss plot were fluctuating within a reasonable boundary to our visual judgements. We also reduced the discount factor to 0.95 from 0.99 in the original paper because our simulation setup had a much smaller state space and much shorter decision horizon.

For the replay memory, the minibatch size was 32, and the size of replay memory was 100,000. The target network update frequency was set to 1,000 for the fixed target network. The learning rate of the network was 0.0001. The training was done on a Windows desktop PC (Intel Core i7-9700K CPU @ 3.60GHz, 32GB RAM and NVIDIA GeForce GTX 1070 Ti) and took approximately 6 days for 4,000,000 episodes (i.e., 4,000,000 point and clicks). All the code was written in `Python`, using the `TensorFlow` library (1.15).

3.4.2 Results. As the reward plot in Fig 10 shows, the simulated agent 's reward flattens. This convergence indicates that the optimal policy has been learned.

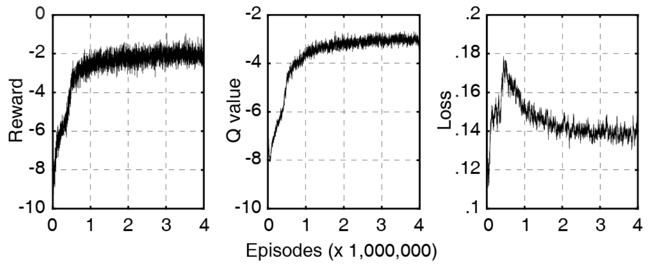

Figure 10: The simulated agent's reward, value function values (Q-value) and training loss.

4 EVALUATION

With the finally obtained simulation model, we conducted three evaluation studies. The first study evaluated the model's point-and-click simulation performance. The output of the simulation was compared with an existing point-and-click user study dataset [53]. The second study observed how the simulated user's behaviour changed when each sub-module was removed (i.e., ablation study). In the third study, participants subjectively evaluated how similar the simulated point-and-click behaviour was to that of the participants in the dataset used in the first study.

We could not find a suitable baseline model to which we could compare our model's performance. The control theoretic models of Muller et al. [48] are the closest as baselines, but they are not intended to simulate point-and-click behaviour on moving targets and do not have an appropriate click mechanism. However, the control theoretical models do not have to be applied only to the stationary target in principle, and if the position and velocity information on the moving target is input into the model, the model can output the trajectory of the cursor regardless. Therefore, in a pilot study, we implemented the second-order lag model ($1/m = 42.97$, $k = 1.07$, $d = 0.23$, $\tau = 0.08$) proposed by Muller et al. in MATLAB Simulink (in which they released the model file) to test whether it can be used as a baseline. In the test, the click action was implemented to execute without failure immediately after the cursor was positioned inside the target. As a result, we confirmed that the output from the model showed a big difference from the performance of the participants in the dataset [53] in terms of trial completion time and cursor trajectory. Therefore, we decided to continue the evaluation without a baseline.

4.1 Validating Simulation Performance

In the previous ICP model study [53], 16 participants (9 males, 7 females) performed the same point-and-click task as in this study (see Section 3.1.2). We received the dataset from the authors and compared the human cursor trajectory and click action with the outputs from our model. In the dataset, participants performed a total of 28,800 point-and-click trials. Among them, 494 trials (1.7 %) were removed due to errors in the logging and a total of 28,306 trials remained. The simulated user of our model also performed the same 28,306 trials. In this process, the simulated user's initial cursor position, target position and target velocity for each trial were set the same as those given to the participants in the ICP study. However, in our model, even after the trial was over, the existing motor plan continued to be executed until the new motor plan was executed, so the initial velocity of the cursor at the beginning of each trial was different from that of the ICP participants. Additionally, since ICP participants performed the trials consecutively without performing them independently, in our model, we assumed that the first BUMP, which begins after a new target is given, omits SA and starts with RP. This is the same as implementing a constant reaction time of T_p (0.1 s).

4.1.1 Results. First, in traditional metrics, our simulation model successfully reproduced the point-and-click performance of ICP participants. The mean trial completion time and click failure rates of ICP participants were 0.89 s (SD=0.45 s) and 37.7 %. For the simulation model they were 0.84 s (SD=0.42 s) and 31.3 %. When trials were binned by the target speed or size, the trial completion time and click failure rate of the simulation model also accurately reproduced those of the ICP participants (see Figure 11). For the distribution of trial completion time and click endpoints (w.r.t. target centre), the output of our model was similar to the distribution of ICP participants (see Figure 12).

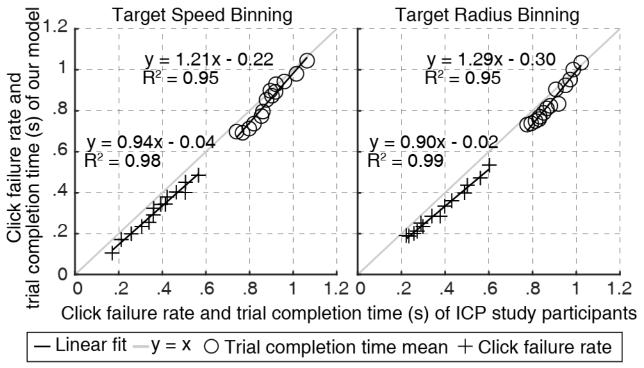

Figure 11: To obtain the mean trial completion time and click failure rate, equal frequency binning of trials (about 2022 trials in a bin) was performed for the target speed and radius. Our simulation model faithfully reproduced the ICP participants' trial completion time and click failure rate.

Cursor trajectories can be evaluated by drawing a phase plot (position vs. velocity) [48]. However, in this study, the target often moved, so drawing a phase plot with the absolute cursor position and velocity would be meaningless. Therefore, we draw phase plots based on the normalised *relative* distance between the cursor and the target. The normalisation was performed by dividing the relative distance in each trial by the initial distance of the target and the cursor. In addition, the point-and-click time was also normalized by dividing it by the completion time of the trial. Figure 13 shows phase plots of our model and ICP participants. The plot shows that our model accurately reproduced the participants' point-and-click trajectory on average overall. However, since our model was simulated assuming the parameter values of an average user, our model could not reproduce the variability between participants. Also, at the beginning of the trial, the simulated user's cursor appears to start moving toward the target *slower* than the participants. We believe that this is because the last BUMP of the previous trial in the simulation continued to run at the beginning of the subsequent trial, and the simulated user had the constant reaction time of 0.1 s. On the other hand, immediately after clicking, the ICP participants may have started moving the cursor in anticipation of the next trial or moved the cursor to a neutral position, such as the center of the screen, which is advantageous on average in obtaining a randomly given target.

In Figure 14, we visualise how our simulated user determined action variables (T_h and K) while performing 28,306 point-and-click trials in this evaluation study. The mean prediction horizon T_h increases up to 2 s at the start of the trial, then immediately decreases to 1.5 s before slowly decreasing to 0.5 s and then rapidly decreasing just before the click. The increase in T_h at the beginning of the trial is because the user waits for the target to bounce off the wall. This leads to higher rewards by reducing the penalty from the movement effort. For the same reason, if the target is small or moves fast, the T_h value is kept higher in the intermediate stage of tracking. At the end of the tracking movement, reducing T_h is an essential strategy to successfully acquire moving targets that do not always stay in the same position. Note that the graph was

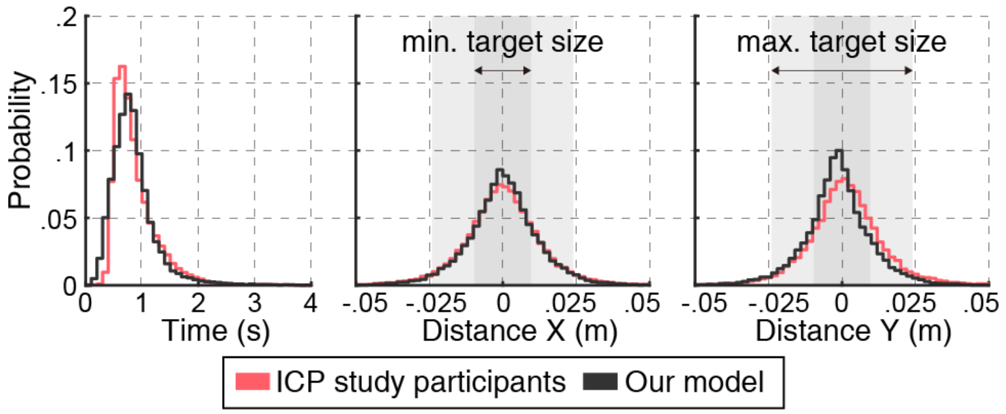

Figure 12: For the distribution of trial completion time and click end point, the simulation successfully reproduced the ICP participants' performance.

drawn on the time axis normalised to the trial completion time (equal frequency binning with about 2,000 trials).

Meanwhile, the click action was mostly executed only when the click difficulty (D_{click}) was at least 3 bits or less (see Figure 14, right). However, 3 bits is only the minimum criterion for clicking, and most of the clicks were actually executed when the difficulty was about 1.1 bit. The user's click failure rate was expected to be approximately 5 % with a difficulty of 1 bit [43]. This is similar to the click failure rate observed in traditional point-and-click tasks (typically 4 %).

The simulated user's click strategy can change if the reward setting is different. For example, if the reward for click success is higher (now R^+=14), the user will try to click in a safer situation with a lower D_{click}. However, the click failure rate cannot be zero at any reward setting due to visual noise, motor noise, and mouse coordinate system disturbance. In fact, when the value of D_{click} was less than 1 bit, the probability of the simulated user executing the click action was drastically reduced. This phenomenon should be interpreted not as the user intentionally trying to execute a click action in a difficult situation but as a result of the user finding it difficult to perform a tracking movement that enables easy click planning. There was no significant difference in the click strategy depending on the speed or size of the target.

4.2 Ablation Study

Our simulation model was a combination of several sub-modules. We could conduct a so-called ablation study by observing how the performance changed when each component was removed to see how these changes contributed to the overall model performance. Without changing the reward setting, we obtained a simulation model for each of the following cases using the same DRL method: (1) w/o motor noise, (2) w/o visual noise, (3) w/o mouse coordinate disturbance and (4) w/o mouse acceleration function. After 4 million training episodes, we confirmed that the policy of action for each model with ablation successfully converged. We also tested how poor the performance of the model became when action variables (T_h and K) were simply fixed without optimisation through DRL (i.e., fixed-action user). For each simulated user with ablation, we performed the same evaluation study that was conducted for the full model.

Other removals were straightforward, but the implementation of the no acceleration condition and of the fixed-action condition need to be explained. No acceleration means that the cursor speed is obtained by multiplying the user's hand speed by a constant gain value (i.e. uniform gain function) [15]. We determined the gain value by dividing the mean cursor speed by the mean hand speed observed in the full model evaluation (1.904).

For the fixed-action condition, T_h was set as the average of all T_h values observed in the full model evaluation (T_h=1.1 s). However, the click decision (K) is difficult to be fixed with either 0 or 1 because it is too artificial to always click or not always click regardless of the given situation. Instead, we implemented the click to be executed when the difficulty of click planning (D_{click}) falls below a certain threshold. When the full model is being simulated, D_{click} for each BUMP can be calculated. Among them, only the BUMPs for which the click action was executed were selected, and their D_{click} values were averaged. The averaged value was 1.53 bits, and in the fixed-action model, when the D_{click} value in a specific BUMP was lower than that, the click action was executed.

4.2.1 Results. First, we could confirm that the simulated user's performance in the fixed-action condition, which did not have the optimised policy of action, was greatly degraded. The mean trial completion time and click failure rate were 1.43 s (SD=0.35 s) and 75.5 %. This is an increase of 0.55 s and 37.8 %p compared to the ICP participants. According to the phase plot (Figure 15), the user with the fixed-action has not succeeded in sufficiently reducing the distance between the target and the cursor and appeared to have clicked at the cost of click failure to prevent the increased penalty from the motor effort.

In the absence of visual noise, interesting results were obtained. The click failure rate was significantly lowered (22.4 %), but the trial completion time slightly increased (M=1.08 s, SD=0.44 s). If there was no visual noise, the user could perfectly estimate the velocity of the target and thus build a motor plan that could more accurately allow the cursor to reach the target centre. This provided the simulated user with a much easier situation when planning for

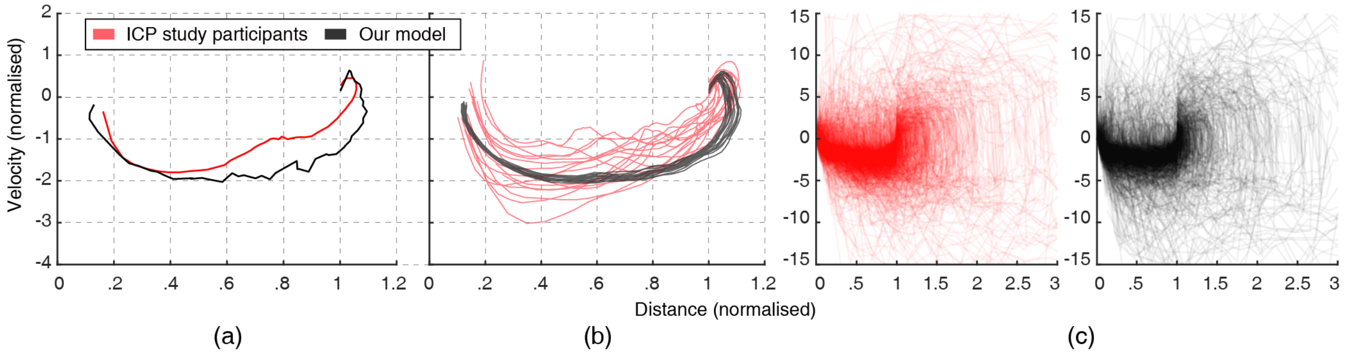

Figure 13: This is the phase plot of the relative motion between the cursor and the target. The distance was normalised by dividing it by the initial distance, and the time was normalised by dividing it by the trial completion time. The velocity was obtained by differentiating the normalised distance. Description of each subplot: (a) average trajectory of all trials performed by ICP participants and the simulated user (note that the simulated user performed the same trials as given to ICP participants), (b) average trajectories for each participant and average trajectories when the simulated user performed the same trials as each participant did, (c) raw trajectories of all trials performed by participant #16 and raw trajectories when the simulated user performed the same trials (see supplementary material for other participant cases). The simulation faithfully reproduces the trajectory of ICP participants on average and the trajectory variability within a participant. However, since the simulated user only implemented the average user, the variability between ICP participants could not be reproduced. Also, the cursor trajectory at the beginning of the trial was slightly different from that of the participants due to the assumption of the model that the last BUMP of the previous trial will continue to be executed in the subsequent trial.

a successful click. Then, the user could devote more effort to target tracking because the user knew that doing enough target tracking can create better click planning situations. This would increase the expected value of the total reward while increasing the user's trial completion time.

In the case of the model with no mouse acceleration, the mean trial completion time and click failure rate were 1.18 s (SD=0.71 s) and 33.9 %, respectively. Compared to the ICP participants, the click failure rate slightly decreased by 3.76 %p, and the trial completion time increased significantly by 0.29 s. This replicated the results of previous studies; when the acceleration function was not applied, the user's point-and-click time increased, and the click failure rate

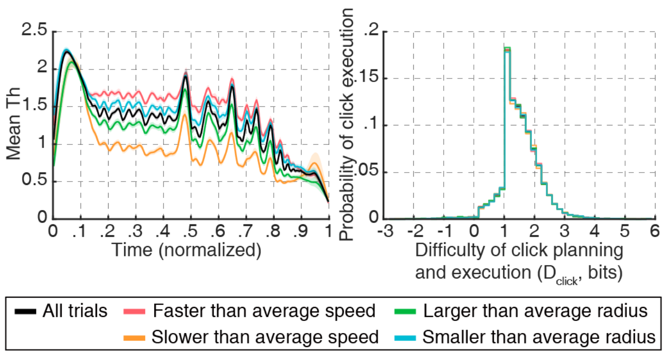

Figure 14: The simulated user's action variables observed in the evaluation study: (1) mean prediction horizon T_h as a function of normalised time (left) and (2) distribution of click planning difficulties (D_{click}) observed in all the BUMPs where the click action was executed, that is, K=1 (right).

remained similar [15, 42]. Typical mouse acceleration functions have a significant effect on the cursor movement only when the hand speed is relatively fast [14]. Therefore, we interpreted from this result that the acceleration function reduced the time required for the tracking movement toward the target but had little effect on the brief movement just before the click. We could also confirm this in the phase plot (see Figure 15). When the mouse acceleration function was removed, the user could not quickly catch up with the target at the beginning of the trial.

In the absence of motor noise, user performance is improved compared to that of ICP participants. The mean trial completion time and click failure rate were 0.78 s (SD=0.38 s) and 27.1 %, respectively. This is an improvement of 0.11 s and 10.6 %p compared to that of the ICP participants. When there was no coordinate disturbance, the mean trial completion time was 0.9 s (SD=0.58 s), and the click failure rate was 35.6 %, showing almost similar performance to the full model. According to the previous study [40], coordinate disturbance affects the performance of users in tasks where the trajectory of a cursor must be precise, such as when drawing. Since point-and-click is not such a task, we interpreted that the absence of coordinate disturbance did not result in a significant performance difference.

In the case of no visual noise, we found interesting results in the plot (see Figure 15). When the trials were binned by the target speed, the trial completion time of the user without visual noise *decreased* as the target speed increased. In our interpretation, this is because as the target speed increased, the motor noise included in the user's movement caught up with the target increases, so it became difficult to plan a click even without visual noise. Therefore, if the target speed is high, the simulated user goes back to the old

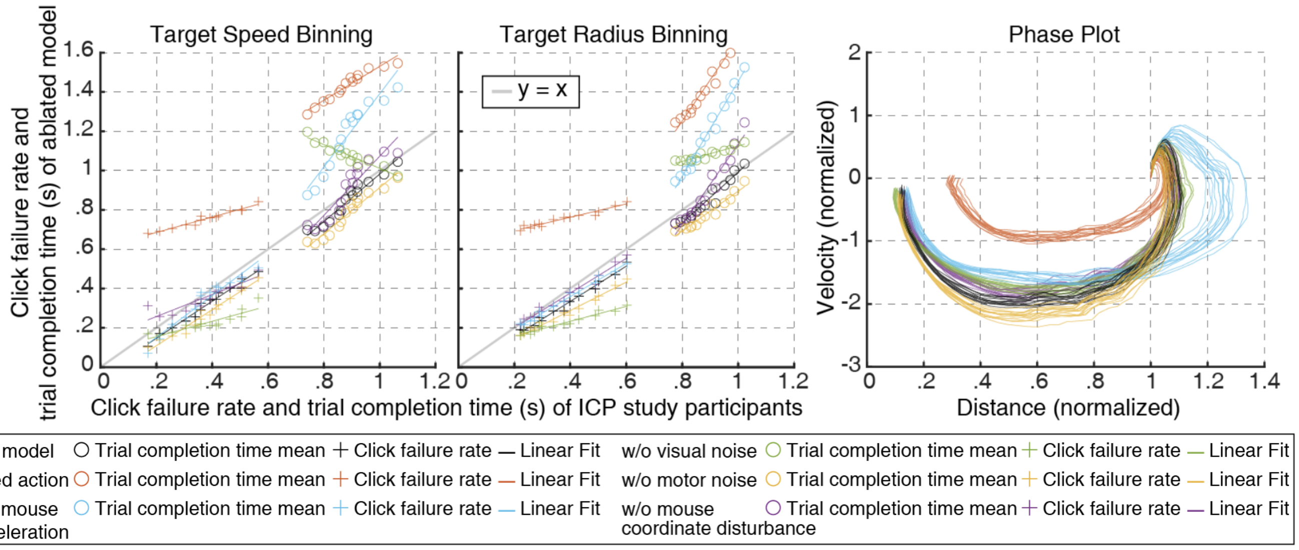

Figure 15: We examined how the model's performance changed when a specific component was removed from the full simulation model.

strategy, thereby reducing the effort spent on tracking movements and executing clicks at risk, even if it is dangerous.

4.3 Subjective Evaluation of Trajectories

We wondered if people could visually distinguish between the point-and-click behaviour that our model outputs and the point-and-click behaviour of real users. As such, we prepared real-time videos of our simulated user and ICP participants performing 28,306 point-and-click conditions on the ICP dataset. Ten people were shown random pairs of 100 videos of them and asked which of them were like real human behaviour. On average, people could hardly differentiate between the output of our model and the behaviour of ICP participants (57.9% of mean success rate with a 12.31% standard deviation). The best participant showed a success rate of 76%. After the experiment, a participant commented that some of the trials ended so quickly, making it difficult to judge. However, from further analysis, we did not find any significant difference in the success rate of participants between trials with a short completion time and trials with a longer completion time. Figure 16 shows the trajectory of the target and the cursor in some of the point-and-click videos the participants saw. The trajectory was not actually visible to the participants.

5 DISCUSSION

For decades, point-and-click has been the most important task of HCI. However, our understanding of user's point-and-click performance has not deepened significantly since the era when studies on Fitts' law were intensively conducted. Through this study, our understanding of point-and-click behaviour can be advanced in two aspects.

First, we demonstrate that the point-and-click behaviour of users, which Fitts' law attempts to describe with just two free parameters, is actually a combination of various sub-processes, such as intermittent motor control, click decision-making, visual perception, signal-dependent motor noise, input sensing, upper limb kinematics, and etc. Such bottom-up or mechanistic modelling not only enables more realistic simulations but also allows us to make imaginative interventions to the system. For example, we demonstrated in the ablation study how the user's performance changed when there was no mouse acceleration function or visual noise. This means that we can provide better answers to *w-questions* (i.e., what-if-things-had-been-different questions) to control and improve the interaction phenomenon [18].

Second, we demonstrated that to realistically simulate the user's behaviour, it is not enough to simply combine various sub-processes and that a consideration of the optimality of the user's action is essential. We found that considering the optimality of the user's action is not simply finding a fixed optimal action variable but rather finding the optimal action policy as a total set of actions that respond to the state of the external environment. This is an idea that has already been proposed in optimal feedback control theory [63] and reinforcement learning studies [16], but our study is the first to apply it to the simulation of point-and-click behaviour.

Despite these contributions, we realise that some assumptions of the model's internal mechanisms have been newly presented in this study and have not been separately validated. Such assumptions include noise-free perception of the position of the target and the cursor at the start of RP, the user's linear extrapolation to the target future position and the process of converting the cursor motor plan to a hand motor plan. We are also aware of the limitations that the applicability of the model has not been explored and that the different variations of MDP formulations (e.g., extreme reward settings or fixed action variables within a trial) have not been compared and tested. By releasing models and datasets, we hope these limitations will be discussed and supplemented through future studies.

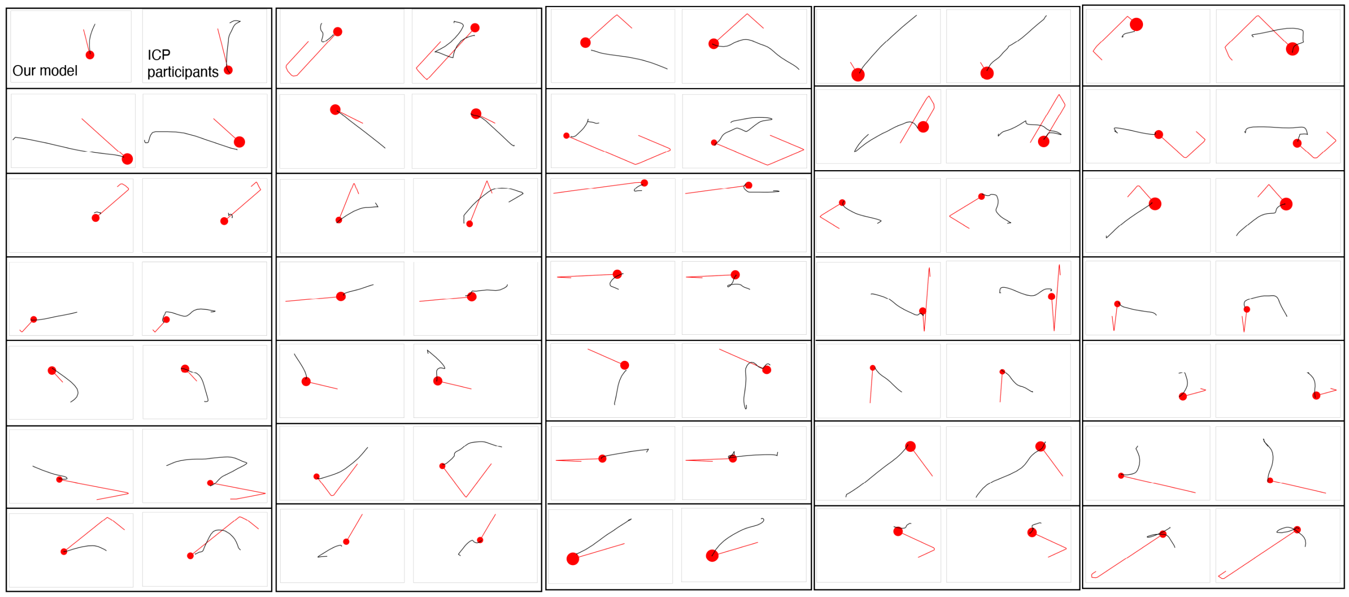

Figure 16: From the moment the target is given to the moment the click occurs, the cursor trajectory simulated by our model (left of each column) and the real user's cursor trajectory (right of each column) were drawn. In the subjective evaluation study using the video with the trajectory removed, the participants hardly distinguished between the two trajectories (red: target, black: cursor)

6 FUTURE WORK

The simulation model proposed in this study reproduces the cognitive-behavioural mechanisms necessary for humans to perform point-and-click tasks as realistically as possible. As a result, the model can accurately predict the traditional aggregated measures of point-and-click performance, such as trial completion time or click failure rate. In addition, the model accurately reproduces the cursor trajectory and the distribution of click endpoints regardless of whether the target is stationary or moving. We hope that this study will not end as a single trial but will contribute to opening a new chapter in HCI modelling research. To this end, we present the following five opportunities for the model to improve in the future.

(1) Perceiving complex target movement: In this study, we only dealt with cases in which the user could easily perceive the movement pattern of the target (i.e., movement at a constant velocity). However, when the target movement became complicated, the time required for point-and-click increased significantly [59]. By referring to past studies on moving target interception [23] or coincidence-anticipation [64], the visual perception module of our model can be improved to be applied to more complex target movements.

(2) Role of eye gaze: According to previous studies [31, 32, 39], the movement of the user's gaze has a significant effect on the performance of information encoding from given targets. For example, when multiple targets are given, the user should recognise the target to be acquired as soon as possible by moving their gaze in an optimal manner. In the future, the EMMA model for simulation of gaze movement [55], drift-diffusion model [39] and foveal vision model [25] for simulation of user's visual encoding process should be considered in model implementation.

(3) Reward function: Depending on the reward setting, the results of the reinforcement learning vary greatly. However, this study only validated the model for a single-reward setting. In the future, it is necessary to explore how the simulated user's behaviour changes in a special reward setting, such as when the penalty for failure to acquire a target is very high. Additionally, through inverse reinforcement learning [1], we could be able to determine the reward settings from expert users' point-and-click data.

(4) Preparation for the next trial: Our model successfully simulated the user's point-and-click behaviour in a single trial. However, when consecutive trials are given, how the user prepares for the newly given target has not been sufficiently considered. Rather than simply reacting to the next target, the user may choose a special strategy, such as anticipating where the target will appear or moving the cursor to a neutral position that is easier on average to acquire targets. For a more realistic simulation, in the future the model need to implement such an advanced input preparation mechanism that runs across trials.

(5) Simulation of individual user behaviour: In this study, an average user was simulated. However, it would be an interesting future study to investigate whether it is possible to reproduce the behaviour of individual users by adjusting model parameters. If it proves possible, a more general simulation of point-and-click behaviour, including end-user variability, can be implemented using parameter distributions measured at the population level for multiple users.

Simulating the behaviour of users with special cognitive and motor characteristics, such as seniors and e-sports athletes, would also be interesting. In this regard, previous studies have reported that the BUMP model can account for 10 Hz physiological tremors of humans [12], and that the parameters of the ICP model can reveal significant differences in cognitive characteristics between gamers and non-gamers [53].

ACKNOWLEDGMENTS

This research was funded by National Research Foundation of Korea (2020R1A2C4002146) and Korea Creative Content Agency (R2019020010). We thank Julien Gori for discussions.

REFERENCES

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

[2] 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] John Robert Anderson. 1996. *The architecture of cognition.* Vol. 5. Psychology Press.

[4] Stanislav Aranovskiy, Rosane Ushirobira, Denis Efimov, and Géry Casiez. 2020. A switched dynamic model for pointing tasks with a computer mouse. *Asian Journal of Control* 22, 4 (2020), 1387–1400.

[5] Myroslav Bachynskyi and Jörg Müller. 2020. Dynamics of Aimed Mid-air Movements. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–12.

[6] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. Menuoptimizer: Interactive optimization of menu systems. In *Proceedings of the 26th annual ACM symposium on User interface software and technology.* 331–342.

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

[8] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: modeling finger touch with fitts' law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 1363–1372.

[9] Xiaojun Bi and Shumin Zhai. 2016. Predicting finger-touch accuracy based on the dual Gaussian distribution model. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology.* 313–319.

[10] Robin Trulssen Bye. 2009. *The BUMP model of response planning.* Ph.D. Dissertation. Sydney, Australia: The University of New South Wales.

[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] Robin T Bye and Peter D Neilson. 2010. The BUMP model of response planning: intermittent predictive control accounts for 10 Hz physiological tremor. *Human movement science* 29, 5 (2010), 713–736.

[13] Stuartk Card, THOMASP MORAN, and Allen Newell. 1986. The model human processor- An engineering model of human performance. *Handbook of perception and human performance.* 2, 45–1 (1986).

[14] 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.

[15] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The impact of control-display gain on user performance in pointing tasks. *Human–computer interaction* 23, 3 (2008), 215–250.

[16] 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.

[17] Hsieh-Ching Chena, Yu-Wen Chena, Yung-Ping Liub, and Yi-Tsong Pan. 2014. Quantitative Assessment of Computer Inputs and Musculoskeletal Complaints among Three Workgroups. *Advances in Physical Ergonomics and Human Factors: Part I* 14 (2014), 375.

[18] Carl F Craver. 2006. When mechanistic models explain. *Synthese* 153, 3 (2006), 355–376.

[19] Holk Cruse, Erhard Wischmeyer, Michael Brüwer, Peter Brockfeld, and A Dress. 1990. On the cost functions for the control of the human arm movement. *Biological cybernetics* 62, 6 (1990), 519–528.

[20] A Diamond and OE Holland. 2014. Reaching control of a full-torso, modelled musculoskeletal robot using muscle synergies emergent under reinforcement learning. *Bioinspiration & biomimetics* 9, 1 (2014), 016015.

[21] Marc O Ernst and Martin S Banks. 2002. Humans integrate visual and haptic information in a statistically optimal fashion. *Nature* 415, 6870 (2002), 429–433.

[22] Andrew H Fagg, Andrew G Barto, and James C Houk. 1998. Learning to reach via corrective movements. In *Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems.* 179–185.

[23] Brett R Fajen and William H Warren. 2007. Behavioral dynamics of intercepting a moving target. *Experimental Brain Research* 180, 2 (2007), 303–319.

[24] Florian Fischer, Arthur Fleig, Markus Klar, Lars Grüne, and Joerg Mueller. 2020. An Optimal Control Model of Mouse Pointing Using the LQR. (2020). arXiv:2002.11596

[25] Luc Florack. 2007. Modeling foveal vision. In *International Conference on Scale Space and Variational Methods in Computer Vision.* Springer, 919–928.

[26] John Gibbon, Russell M Church, Warren H Meck, et al. 1984. Scalar timing in memory. *Annals of the New York Academy of sciences* 423, 1 (1984), 52–77.

[27] Jin Huang, Feng Tian, Nianlong Li, and Xiangmin Fan. 2019. Modeling the Uncertainty in 2D Moving Target Selection. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology.* 1031–1043.

[28] Jun Izawa, Toshiyuki Kondo, and Koji Ito. 2004. Biological arm motion through reinforcement learning. *Biological cybernetics* 91, 1 (2004), 10–22.

[29] Wolfgang Jaschinski. 2002. The proximity-fixation-disparity curve and the preferred viewing distance at a visual display as an indicator of near vision fatigue. *Optometry and Vision Science* 79, 3 (2002), 158–169.

[30] Bonnie E John and David E Kieras. 1996. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)* 3, 4 (1996), 320–351.

[31] 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.

[32] 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.

[33] Hiroyuki Kambara, Kyoungsik Kim, Duk Shin, Makoto Sato, and Yasuharu Koike. 2009. Learning and generation of goal-directed arm reaching from scratch. *Neural Networks* 22, 4 (2009), 348–361.

[34] Andreas Karrenbauer and Antti Oulasvirta. 2014. Improvements to keyboard optimization with integer programming. In *Proceedings of the 27th annual ACM symposium on User interface software and technology.* 621–626.

[35] David Kieras et al. 2001. Using the keystroke-level model to estimate execution times. *University of Michigan* 555 (2001).

[36] Sunjun Kim, Byungjoo Lee, Thomas van Gemert, and Antti Oulasvirta. 2020. Optimal Sensor Position for a Computer Mouse. (2020). arXiv:2001.03352

[37] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. (2014). arXiv:1412.6980

[38] Konrad P Körding and Daniel M Wolpert. 2006. Bayesian decision theory in sensorimotor control. *Trends in cognitive sciences* 10, 7 (2006), 319–326.

[39] Ian Krajbich and Antonio Rangel. 2011. Multialternative drift-diffusion model predicts the relationship between visual fixations and choice in value-based decisions. *Proceedings of the National Academy of Sciences* 108, 33 (2011), 13852–13857.

[40] Byungjoo Lee and Hyunwoo Bang. 2015. A mouse with two optical sensors that eliminates coordinate disturbance during skilled strokes. *Human–Computer Interaction* 30, 2 (2015), 122–155.

[41] 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.

[42] Byungjoo Lee, Mathieu Nancel, Sunjun Kim, and Antti Oulasvirta. 2020. AutoGain: Gain Function Adaptation with Submovement Efficiency Optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'20). ACM, New York, NY, USA. DOI: http://dx. doi. org/10.1145/3313831.3376244.*

[43] 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.

[44] 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.

[45] Duane T McRuer and Ezra S Krendel. 1974. *Mathematical models of human pilot behavior.* Technical Report. ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE).

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

[47] 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.

[48] Jörg Müller, Antti Oulasvirta, and Roderick Murray-Smith. 2017. Control theoretic models of pointing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 24, 4 (2017), 1–36.

[49] Robert M Nosofsky. 2011. The generalized context model: An exemplar model of classification. *Formal approaches in categorization* (2011), 18–39.

[50] 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.

[51] 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.

[52] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2765–2774.

[53] Eunji Park and Byungjoo Lee. 2020. An Intermittent Click Planning Model. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'20). ACM, New York, NY, USA. DOI: http://dx. doi. org/10.1145/3313831.3376725*.

[54] Ashutosh B Potdar, Pratapsingh Rathod, Pallavi A Potdar, and Manjiri M Desai. 2019. A Study of Estimation of Stature from Forearm Length. *Indian Journal of Forensic Medicine & Toxicology* 13, 2 (2019), 219–221.

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

[56] 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.

[57] Marilyn C Smith. 1967. Theories of the psychological refractory period. *Psychological bulletin* 67, 3 (1967), 202.

[58] 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.

[59] Josef Spjut, Ben Boudaoud, Kamran Binaee, Jonghyun Kim, Alexander Majercik, Morgan McGuire, David Luebke, and Joohwan Kim. 2019. Latency of 30 ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz. In *SIGGRAPH Asia 2019 Technical Briefs*. 110–113.

[60] 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.

[61] Kashyap Todi, Jussi Jokinen, Kris Luyten, and Antti Oulasvirta. 2018. Familiarisation: Restructuring layouts with visual learning models. In *23rd International Conference on Intelligent User Interfaces*. 547–558.

[62] Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and explore with a layout optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 543–555.

[63] Emanuel Todorov and Michael I Jordan. 2002. Optimal feedback control as a theory of motor coordination. *Nature neuroscience* 5, 11 (2002), 1226–1235.

[64] James R Tresilian. 2005. Hitting a moving target: perception and action in the timing of rapid interceptions. *Perception & Psychophysics* 67, 1 (2005), 129–149.

[65] Julia Trommershauser, Konrad Kording, and Michael S Landy. 2011. *Sensory cue integration*. Oxford University Press.

[66] Zhou Wang and Qiang Li. 2007. Video quality assessment using a statistical model of human visual speed perception. *JOSA A* 24, 12 (2007), B61–B69.

[67] 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.

[68] Charles J Worringham and Dennis B Beringer. 1998. Directional stimulus-response compatibility: A test of three alternative principles. *Ergonomics* 41, 6 (1998), 864–880.

[69] Shumin Zhai. 2004. Characterizing computer input with Fitts' law parameters—the information and non-information aspects of pointing. *International Journal of Human-Computer Studies* 61, 6 (2004), 791–809.

[70] Shumin Zhai, Michael Hunter, and Barton A Smith. 2002. Performance optimization of virtual keyboards. *Human–Computer Interaction* 17, 2-3 (2002), 229–269.