Optimizing the Timing of Intelligent Suggestion in Virtual Reality

Difeng Yu
University of Melbourne
Melbourne, VIC, Australia

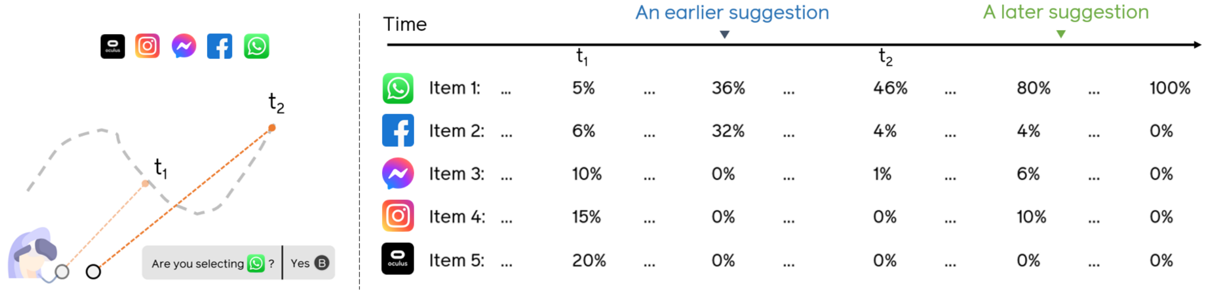
Ruta Desai
Reality Labs Research
Seattle, WA, USA

Ting Zhang
Reality Labs Research
Seattle, WA, USA

Hrvoje Benko
Reality Labs Research
Seattle, WA, USA

Tanya R. Jonker
Reality Labs Research
Seattle, WA, USA

Aakar Gupta
Reality Labs Research
Seattle, WA, USA



		t_1				t_2				
Item 1:	...	5%	...	36%	...	46%	...	80%	...	100%
Item 2:	...	6%	...	32%	...	4%	...	4%	...	0%
Item 3:	...	10%	...	0%	...	1%	...	6%	...	0%
Item 4:	...	15%	...	0%	...	0%	...	10%	...	0%
Item 5:	...	20%	...	0%	...	0%	...	0%	...	0%

Figure 1: An overview of the intelligent suggestion timing problem. While a user is attempting to select an icon in virtual reality, a target prediction model could be continuously estimating the likelihood that the user will select each icon (e.g., at timestamp t_x and t_y). Depending on the results of these estimations, a system could then display an intelligent suggestion to the user that highlights the most probable icon for them to select. This suggestion, for example, could enable them to select an icon using a simple click, so that the user does not need to manually point towards the icon. While such suggestions could improve the usability of intelligent user interfaces, it is currently unknown whether early suggestions, which could save the user time and effort but may be less accurate, or later suggestions, which could save less time and effort but may be more accurate, are more beneficial for users.

ABSTRACT

Intelligent suggestion techniques can enable low-friction selection-based input within virtual or augmented reality (VR/AR) systems. Such techniques leverage probability estimates from a target prediction model to provide users with an easy-to-use method to select the most probable target in an environment. For example, a system could highlight the predicted target and enable a user to select it with a simple click. However, as the probability estimates can be made at any time, it is unclear *when* an intelligent suggestion should be presented. Earlier suggestions could save a user time and effort but be less accurate. Later suggestions, on the other hand, could be more accurate but save less time and effort. This paper thus proposes a computational framework that can be used to determine the optimal timing of intelligent suggestions based on user-centric costs and benefits. A series of studies demonstrated the value of the framework for minimizing task completion time and maximizing suggestion usage and showed that it was both theoretically and empirically effective at determining the optimal timing for intelligent suggestions.

CCS CONCEPTS

• **Human-centered computing → HCI theory, concepts and models**; *Mixed / augmented reality*; *Virtual reality*.

KEYWORDS

intention prediction, intelligent interfaces, optimization framework, reinforcement learning

ACM Reference Format:
Difeng Yu, Ruta Desai, Ting Zhang, Hrvoje Benko, Tanya R. Jonker, and Aakar Gupta. 2022. Optimizing the Timing of Intelligent Suggestion in Virtual Reality. In *Woodstock '22: ACM Symposium on Neural Gaze Detection, June 03–05, 2022, Woodstock, NY*. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Target selection in virtual and augmented reality (VR/AR) systems is difficult, especially when interaction scenarios are complex (e.g., with small, faraway, cluttered objects) and input techniques are cumbersome to use (e.g., mid-air hand pointing). Recent research has utilized statistical or machine learning models to estimate the likelihood of a user selecting different items or objects of interest [20, 25, 65]. Based on the estimated probabilities computed by these models, an interaction system may then use visual highlighting or display a notification to draw the user's attention towards the most probable target. Next, a user may select the predicted target with a shortcut (such as a simple click) [1, 30, 70]. Such techniques can alleviate the need to manually point at targets or conduct a full visual search of an environment, potentially leading to quicker, easier, and more comfortable interactions. They can also be useful within VR/AR systems that employ noisy, high-friction input modalities [1, 22, 70] or support scenarios that require users to complete manually-intensive or mental-demanding tasks, such as selecting objects in a cluttered environment or navigating through a complex hierarchical menu [17, 25, 38, 71].

While current target prediction models can determine *which* target a user may select, they cannot determine *when* intelligent suggestions should be provided to users. While an earlier suggestion could save a user time and effort, such suggestions have a higher chance of being incorrect, which could cause users frustration, break their trust, or decrease their performance [12, 40]. On the other hand, later suggestions are likely to be more accurate but less beneficial because users have already spent ample time and effort to complete their task. By the time a model has accumulated enough evidence to be certain of a user's intended target, the user may have almost completed their action, thus rendering the late-breaking intelligent suggestion useless or disruptive (refer to Figure 1 for a problem overview).

Despite this important nuance, existing target prediction models have not scrutinized *when* to offer a suggestion and instead used a heuristically proposed probability threshold. For example, prior work on forecasting which target a user might reach towards with their hands used a threshold of 85% because the model seemed accurate enough at that point based on their evaluation of the model confidence value over time [20]. In contrast, Huang et al. used a threshold of 43% when predicting which sandwich ingredient a user might choose via gazing [36]. They used this threshold because it was based on the average model confidence value for a correct prediction. The mixture of design intuitions and model performance observations used in this prior work may not lead to optimal suggestion timings—one may wonder if a better threshold could be chosen. Furthermore, this prior research did not consider the user-centric costs and benefits of intelligent suggestions (e.g., the exact time saved by a suggestion). Thus, this research introduces the *COBO* (cost-benefit optimization) framework, which determines the optimal timing of intelligent suggestions by considering user-centric costs and benefits. Specifically, COBO uses the probability estimates computed by a target prediction model over time as input and quantifies the cost and benefit of a suggestion to produce a final gain function. The obtained gain function then enables the determination of the most beneficial timing for suggestions either through optimization of this function or through designer's intuition.

To study how users would respond to an intelligent suggestion displayed at different timings, a dense target selection task and a text matching task were implemented in VR. VR was chosen as the testbed because VR input techniques such as mid-air pointing are effortful and are likely to benefit from intelligent suggestions. Based on the study results, cost and benefit functions were developed and simulations were run under two optimization strategies – Optimal Thresholding and Reinforcement Learning – to minimize user task completion time and maximize intelligent suggestion usage. The efficacy of these strategies was then verified in two validation experiments, which showed that COBO was helpful for determining the optimal timing of intelligent suggestions both theoretically and empirically.

The primary contributions of this research are:

- A framework (i.e., *COBO*) to optimize the timing of intelligent suggestions through a computational approach that considers user-centric costs and benefits.
- Study outcomes that demonstrate the effectiveness of COBO for intelligent suggestion timing optimization on two objectives: minimizing user task completion time and maximizing intelligent suggestion usage.

2 BACKGROUND AND RELATED WORK

This research was informed by facilitation techniques that aim to improve user performance and save user efforts in object selection tasks. It also took inspiration from works that applied probabilistic models to estimate user-intended target(s) and research that leveraged Reinforcement Learning for objective optimization in interactive applications.

2.1 Selection Facilitation Techniques

Selection facilitation techniques have been used as a method to improve interaction since the introduction of early graphical user interfaces. While numerous techniques have been proposed, the majority decrease the movement distance required to reach a target and/or increase the effective size of the target [28]. To shorten the movement distance, techniques may snap the cursor to the target (e.g., [10, 73]). To increase the target size, techniques may expand the target [44] or resize the cursor [28, 46]. A visual indicator (e.g., visual highlighting) may also provide feedback when a technique has selected a candidate object. The user can then use an explicit action (e.g., a button press) to confirm that the object that is currently selected is the one they desired to select.

Selection facilitation techniques have also been explored in VR/AR scenarios (see surveys such as [6, 42]). For example, Schjerlund et al. applied multiple virtual hands to shorten the selection distance [60] and Baloup et al. compared various raycasting-based methods that enlarged the objects' effective size in VR [11]. Selection facilitation techniques have been applied to VR/AR systems because mid-air pointing, which is a commonly used input modality in these systems for 3D input, can be inefficient and imprecise [7, 70].

More relevant to the present research are selection techniques that *predict* user-intended targets [1, 70]. In addition to decreasing target distances and increasing target sizes, prediction-based methods have also been found to reduce search time [15]. While a user may have trouble finding the intended target in more complex environments (e.g., those with lots of visual clutter), an intelligent suggestion can present a potential target to users, thus minimizing the time spent searching and manually pointing. We describe these techniques in the next section.

2.2 Target Prediction

Users' intended selection targets can be sensed through behavioral cues, such as body and eye movements. Much existing research focuses on building models that appropriate gaze traces or scanpaths to predict selection intentions [21, 37, 39, 59, 61, 72]. For example, Borji et al. [15] built models that predicted search targets based on gaze fixations on a large random-dot array. Their modeling rationale was that attention and gaze are guided toward visual features that are similar to a search target. Using this approach, they demonstrated that their models outperformed a random baseline, especially when a larger number of fixations was considered. Huang et al. [36] used a support vector machine model to predict a customer's intended target in a sandwich-making scenario and made correct estimations approximately 1.8 seconds before a customer's spoken request. Sattar et al. [58] proposed a model to predict the categories and attributes of user intended objects from gaze data, which were then used to reconstruct plausible targets. Researchers have also explored target forecasting in VR (e.g., [35]), with some taking advantage of gaze fixations to anticipate users' hand movements while reaching for objects [19, 26].

Hand and input device trajectories have also been used in selection tasks to infer user-intended targets [13, 14, 47, 74]. For example, Ahmad et al. [1–4] investigated probabilistic intent prediction approaches for in-vehicle touchscreen input based on pointing gestures. Yu et al. [70] examined the selection distribution of VR input controllers and used this information to predict the likelihood of a user selecting a candidate object. Clarence et al. [20] used long short-term memory (LSTM) models to predict the probability of selecting candidate objects using hand-reach features such as position and orientation. Researchers have also predicted future cursor positions in target-agnostic manners (e.g., [10, 30–32, 41, 43, 51, 68]).

In addition to user behaviour, models can also make use of users' preceding actions or contextual information to infer their next selection intent [27, 66, 67]. For instance, Goodman et al. [27] applied a language model for text entry to estimate the most likely selected key based on an entered sequence and the current input distribution. White et al. [67] leveraged interaction contexts such as previous search queries and clicks to predict users' short-term interests.

Although target prediction models can be effective at determining *which* object a user intends to select previous work has not examined *when* intelligent suggestion should be enabled to maximize its benefits. Some researchers have used design intuitions to trade-off between successful early predictions and the possibility of introducing false positives [1, 20, 36]. Others chose to always display a predicted target (e.g., typing predictions). However, intuitions may not lead to optimized performance and always-on, constantly changing suggestions during cursor navigation or visual search might lead to user costs that were not anticipated, especially in VR/AR scenarios where screen space is limited and distraction may be costlier. As such, our research introduces a method for optimizing the timing of intelligent suggestions that was designed to be extensible to any of these aforementioned prediction models.

2.3 Reinforcement Learning

Recently, reinforcement learning (RL) has been used in the development of adaptive user interfaces [25, 62] and human behavior simulations [18, 33]. In a typical training setting, an RL agent interacts with its environment using a set of actions and receives corresponding feedback (i.e., rewards or penalties) to help it learn from the environment [8]. Through this trial-and-error process, the agent can discover an action policy that leads to a maximized reward. Such a learning paradigm may be particularly suitable for interactive settings that incorporate human-in-the-loop [9].

HCI researchers have applied both model-based and model-free RL for interface optimization. For example, Todi et al. [62] leveraged model-based RL that utilized predictive HCI models to estimate a potential reward of an agent's action. Their model-based agent learned to adapt menu interfaces through order changing or grouping to improve user performance. In contrast, Gebhardt et al. [25] applied model-free RL to support users in a visual search task by showing and hiding object labels (e.g., price tags). Their RL agent observed user behavior (i.e., gaze trajectories) and received rewards or penalties depending on whether a label was shown when the user's gaze point was fixated on the object. Compared to model-based approaches, the model-free agent did not make predictions about the next state and reward before it took an action.

The present work employs model-free RL to discover an optimal policy of suggestion timing. Model-free RL was chosen because it does not require a transition dynamics model to derive a useful policy. The reward function integrated user-centric costs and benefits in terms of, for example, the exact time saved in seconds.

3 RESEARCH OVERVIEW

Our framework relies on quantifying user-centric costs and benefits of a suggestion over time (e.g, the exact time saved by a suggestion) to produce a final gain function for optimal suggestion timing determination. In the following sections, we introduce our framework and present three studies that aimed to demonstrate and validate the proposed framework.

The first is a user study to collect data to approximate the cost and benefit functions related to two optimization objectives (i.e., time saved and suggestion usage percentage) in a manually-intensive task and a mentally-demanding task. This is essential to complete the cost and benefit quantification step in the framework.

The second is a simulation study where simulations were run with two optimization strategies (Optimal Thresholding and Reinforcement Learning) for single- and multi-objective optimization. These simulations aimed to optimize the gain functions related to the objectives and theoretically evaluate the optimization strategies.

In the third study, the optimization findings were empirically validated by running user studies that compared the optimal timing of intelligent suggestions produced by our framework against two baselines—heuristic-based thresholding and no suggestion. The baselines help contextualize the impact of our solution relative to a literature baseline and interfaces that offer no suggestions.

4 COBO FRAMEWORK

COBO (cost-benefit optimization) is a framework to optimize *when* to display intelligent suggestions by considering the costs and benefits that an intelligent suggestion may provide to the user (e.g., the exact time saved) given specific timing and model probabilities. More precisely, COBO takes input probability estimations from a target prediction model and user-centric costs and benefits of a suggestion over time to form a final gain function. The optimized suggestion timing is then determined by finding the maximum gain on this gain function curve (Figure 2). To apply the COBO framework, three components are needed: a target prediction model, a method for cost and benefit quantification, and a strategy for gain function optimization.

4.1 Target Prediction Model

Target prediction models are probabilistic models that infer a user's intended target of interest. A model typically produces a probability distribution $\{p_t^k\}$ among N potential candidates, which indicates the likelihood of a user selecting each candidate $k \in \mathcal{K} = \{1,...,N\}$ at timestamp t (Figure 2 left). It may then output the most likely target and its corresponding probability value q_t (also called the model confidence). In the model, timestamp $t \in \{1,...,T\}$, where T is the total number of timestamps that the model produces estimations since the onset of the selection until the user manually selects a target. In the present work, the target prediction models produce output at a constant frequency f. Therefore, timestamp t can be converted to time in seconds t_s using $t_s = t/f$.

The target prediction models can be trained using data collected from various information channels (e.g., user hand movement [1, 20], eye gaze information [21, 36], prior selection information [27], etc.). While the output of the target prediction model (i.e., probability estimates over time) is used as input to the COBO framework, the model itself is not a part of the framework. For simplicity, this research only displays intelligent suggestions for the most probable object. Thus, only the model confidence q_t is used as input to the COBO framework rather than the whole probability distribution. It is also assumed that model confidence is a reasonable approximation of the ground truth prediction accuracy [29, 49].

4.2 Cost and Benefit Quantification

COBO requires a quantification of the user-centric costs and benefits of displaying an intelligent suggestion over time based on the optimization objective. For example, if the objective is to minimize user task completion time, the cost and benefit quantification can use an estimation on how long it takes users to respond to suggestions, how much time a correct suggestion may save, and how much of a time delay an incorrect suggestion may cause. Such quantification can be specified from the results of empirical user studies or through literature-informed assumptions. The obtained cost function $\mathtt{Cost}(t)$ and benefit function $\mathtt{Benefit}(t)$ can then be used to build a final gain function.

The total gain of displaying an intelligent suggestion for the most probable object at a particular timestamp t is shown in Equation 1. The gain function is equivalent to the benefit obtained, multiplied by the probability that the predicted object is the true target minus the cost, multiplied by the probability of the object not being the real target.

$$\mathtt{Gain}(t) = \mathtt{Benefit}(t) \cdot q_t - \mathtt{Cost}(t) \cdot (1 - q_t) \tag{1}$$

When applying the COBO framework, the gain objective can vary in different applications according to a designer's needs (e.g., minimizing completion time, minimizing induced errors, maximizing user satisfaction, etc.). This research demonstrates the optimization of two gain objectives, i.e., the time saved by users and the suggestion usage percentage.

4.2.1 *Time Saved by Users.*

Task completion time is an obvious metric of user task performance. Ideally, an effective user interface shortens task completion time, while maintaining accuracy to increase user efficiency. To maximize time savings for users, the following three variables were considered when displaying an intelligent suggestion at timestamp t:

- Response time $\mathtt{RT}(t)$: the time elapsed between the first appearance of a correct suggestion and the time when the user applies the suggestion (e.g., through a simple click).
- Response rate $\mathtt{RR}(t)$: the overall user response rate to a correct suggestion.
- Delayed time $\mathtt{DT}(t)$: the average time delay caused by displaying an incorrect suggestion.

For simplicity, we assume that there are minimal effects of i) the delayed time of a correct suggestion if a user does not apply it and ii) the response time of an incorrect suggestion if a user assumes it is correct.

For a given trial with total timestamps T, the potential benefit of displaying a suggestion at t is represented in Equation 2. The equation can be interpreted as the estimated timestamps saved if a correct suggestion is given at t, multiplied by their rate of response. The $\mathtt{max}$ function ensures the benefit value is no smaller than 0.

$$\mathtt{Benefit}(t) = \mathtt{max}(0, T - (t + \mathtt{RT}(t))) \cdot \mathtt{RR}(t) \tag{2}$$

The potential cost is the time delay caused by an incorrect prediction (Equation 3).

$$\mathtt{Cost}(t) = \mathtt{DT}(t) \tag{3}$$

Inserting Equation 2 and 3 into Equation 1, results in an estimated gain function that considers the timestamps saved for users (Equation 4). It can be converted to the time saved in seconds by dividing it by the model output frequency f.

$$\mathtt{Gain}(t) = \mathtt{max}(0, T - (t + \mathtt{RT}(t))) \cdot \mathtt{RR}(t) \cdot q_t - \mathtt{DT}(t) \cdot (1 - q_t) \tag{4}$$

4.2.2 *Suggestion Usage Percentage.*

Although time savings is a useful objective for performance improvement, it may not necessarily be valuable to the user experience. For example, previous work has shown that even when word prediction may impair average text entry speeds on mobile devices, users still prefer to use them [50, 54]. As such, we also sought to optimize for intelligent suggestion usage percentage. It was assumed that as long as a user applies an intelligent suggestion, it leads to a preferred user experience.

Based on this, the gain function can be written as Equation 5. The benefit function is approximated by the likelihood of users responding to a correct suggestion. For simplicity, the probability of users applying an incorrect suggestion is ignored so the cost function is omitted.

$$\mathtt{Gain}(t) = \mathtt{RR}(t) \cdot q_t \tag{5}$$

4.3 Gain Optimization

The value of the gain function $\mathtt{Gain}$(t) changes over time such that the model confidence value q_t, the user-centric cost $\mathtt{Cost}$(t), and benefit $\mathtt{Benefit}$(t) will be different as the task progresses and t increases. In real applications, the target selection model does not infer when a user starts the task ($t = 0$) or when the user finishes the task, so the task progress is unknown to the prediction model. One solution is thus to infer t from the the real-time model confidence value of the target prediction model q_t because the model tends to become more confident in its predictions as the user reaches the end of their task. Several prior studies have indicated that the relationship between t and q_t may follow a sigmoid function [20, 36], thus the implicit

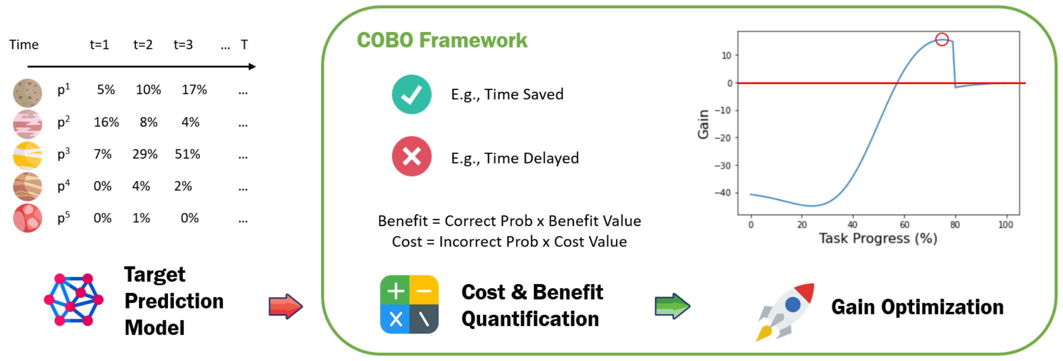


Figure 2: An overview of the COBO framework. COBO uses the probability estimates of a target prediction model as input and quantifies the cost and benefit of the suggestion over time to produce a final gain function. The gain function is computed using the benefit of displaying a suggestion minus its cost across the time axis. By applying an optimization strategy, the framework determines when displaying a suggestion will be useful (gain > 0) and when the gain value (max(gain)) will be maximized.

relationship between t and q_t can be modelled as $t = g(q_t)$. By doing this, the final objective function (Equation 6) only depends on the real-time confidence output q_t. The objective function returns the q_t that leads to the maximum gain. The returned q_t can be directly applied to determine a suggestion timing. For example, if the optimized $q_t = 0.6$, the system should display an intelligent suggestion when the model confidence reaches 0.6.

$$\underset{q_t\in[0,1]}{\texttt{argmax}}\ [\texttt{Benefit}(g(q_t))\cdot q_t - \texttt{Cost}(g(q_t))\cdot(1-q_t)] \qquad (6)$$

In practice, we obtain the mapping function $t = g(q_t)$ from a training dataset D_{train}. The purpose of D_{train} is to provide known relationship between t and q_t so that an optimization strategy can learn how to handle new real-time q_t values. In this work, we created a dataset, D_{train}, wherein each data trial consisted of known q_t values for all $t \in \{1,...,T\}$. Such a dataset can also be generated by using a trained prediction model to produce q_t for each $t \in \{1,...,T\}$ of the feature data (e.g., hand movement [20] or gaze information [36] over time). Once D_{train} and the cost and benefit functions are available, an optimization strategy can calculate the expected gain by simulating the effect of enabling intelligent suggestions at different q_t (which correspond to a known t) on the trials in D_{train}, to consequently compute an optimal solution over the training set. With the hypothesis that the training data is a reasonable approximation of the unseen testing data, the optimized solution can be generalized to real applications.

Since the objective is to find a q_t or a set of q_ts that can lead to the maximum gain, various optimization methods can be applied to solve this problem. In this work, two optimization strategies (i.e., Optimal Thresholding and Reinforcement Learning) were explored.

4.3.1 *Optimal Thresholding (OT).*

The Optimal Thresholding strategy aimed to obtain a single optimized model confidence threshold that worked best on D_{train}. To achieve this aim, different confidence values $q_t \in [0, 1]$ were tested and the q_t that lead to the highest expected gain on D_{train} was selected.

4.3.2 *Reinforcement Learning (RL).*

Rather than relying on a single threshold for all trials, RL-based optimization strategies can provide "dynamic thresholds" based on the profile of each trial (e.g., the speed of increase of the model confidence value). This has the potential to further boost the optimization performance compared to Optimal Thresholding. Therefore, RL was applied to derive optimal policies for intelligent suggestions that could reach the highest gain on D_{train}. Specifically, our RL agents observed the incoming probability estimations and explored different action sequences (i.e., displayed an intelligent suggestion or not) to ultimately find optimal action sequences that would lead to the maximum gain. Additional details about the RL agents are in Section 6.3.

5 STUDY 1 - DATA COLLECTION

The primary goal of the first study was to quantify the cost and benefit of the two optimization objectives. To this end, data was collected from participants while they responded to an intelligent suggestion displayed at different timings. Specifically, this study focused on how much time it took participants to respond to a correct suggestion, the usage percentage of the correct suggestion over time, and the trial completion delay incurred by an incorrect suggestion. Two different task scenarios (manually-intensive vs. mentally-demanding) and two different suggestion types (visual highlighting versus pop-up notification) were used to explore whether these factors would lead to different participant responses. We tested these factors because they could be the main determinants of user behavior towards an intelligent suggestion.

We used a two-session data collection study methodology. In the first session, baseline user performance (e.g., task completion time) was collected while participants performed a dense target selection task and a text matching task. The baseline user performance was used to inform the suggestion timing interval for the second session. In the second session, correct and incorrect suggestions within the earlier determined timing intervals were displayed and the resulting participant behavioral data were recorded. This enabled the measurement of the costs and benefits of the suggestion.

We here prioritize high-level concepts and more relevant contents in our presentation. We refer readers to Appendix A for more detailed descriptions of the task scenarios and suggestion methods and the significance testing results.

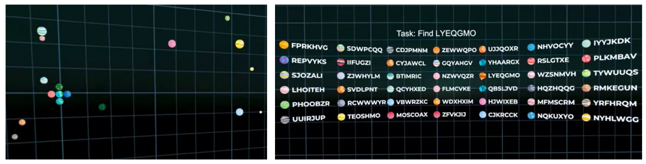


Figure 3: Screenshots of the dense target selection task (left) and the text matching task (right).

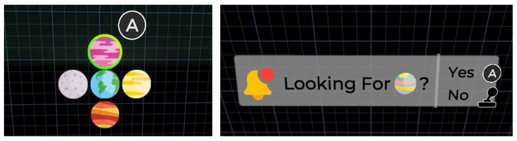


Figure 4: Highlighting notification (left) and pop-up suggestion (right) used in the dense target selection task.

5.1 Participants and Apparatus

Sixteen participants (6 women and 10 men) were recruited and provided informed consent on attending the study. Participant ages ranged from 23 to 47 ($mean = 36.6$, $std = 7.7$, one participant did not report their age). All participants had normal or corrected-to-normal vision and all were right-handed. Twelve participants had used VR devices for 0-5 hours per week, three used them for 5-10 hours, and one had never used a VR device before. As participation was remote, participants received equipment to use in the study by mail (i.e., an Oculus Quest 2, two Touch controllers, and a laptop with an GTX 1070 graphics card) and met with the researchers during a video call to complete the study.

5.2 Task Scenarios

Two task scenarios, representative of common interaction tasks that are effortful to perform, were employed (see Figure 3). The dense target selection task represented a manually-intensive task, where participants needed to select a small object located at the center of a cluster [46, 64]. The text matching task served as a mentally-demanding task, where participants needed to find and select an object with text that matched a target text. This task simulated real-world, search-heavy scenarios like searching for ingredients from a receipt, finding street names on a map, or browsing through a menu [52].

5.3 Suggestion Method

Two suggestion methods were used in the study—a highlighting suggestion and a pop-up suggestion. With the highlighting suggestion, a blinking fluorescent outline was displayed around the suggested object (Figure 4 left). The highlighting suggestion was in-situ, so it remained at the object location without following the direction participants were looking. With the pop-up notification suggestion, a suggestion window appeared at the bottom of the participant's current viewing direction (Figure 4 right) [57]. When participants rotated their viewing direction, the pop-up notification followed the viewing direction. For both suggestions, participants could quickly access the suggested object via the Button A or discard the suggestion by tilting the joystick to the right.

5.4 Study Design

The study included two sessions. The first session used a within-subject design with one factor, Task Type (dense target selection and text matching), to collect baseline user performance. Each task had 48 trials, with the first 3 trials being discarded as practice trials. The order of Task Type was counterbalanced. In total, 1440 trials were recorded (= 16 participants × 2 task types × 45 repetitions).

The second session was conducted on a later day with the same pool of participants after they had all finished the first session. It also used a within-subject design but had three factors: Task Type (dense target selection and text matching), Suggestion Method (highlighting and pop-up notification), and Suggestion Mode (correct, incorrect, and no suggestion). A suggestion, if there was one, was generated within a specific timing interval ($[0s, 3.1s]$ for the dense target selection task and $[0s, 7.6s]$ for the text matching task). The suggestion timing was then randomly sampled within this interval in each task to help us better understand how users respond to suggestions over time. The mean task completion time from the first session was used as the maximum suggestion timing for the second session, as users normally finish the task manually before these upper-bound times. The order of Task Type and Suggestion Method were counterbalanced, and the order of Suggestion Mode was randomized within each block. When a participant was working on a certain task type with a suggestion method, a suggestion may or may not appear and could be correct or incorrect. In Session 2, each condition was repeated 32 times (2 practice trials). In total, 5760 trials were recorded (= 16 participants × 2 task types × 2 suggestion methods × 3 suggestion modes × 30 repetitions).

5.5 Study Procedure

The same procedure was used for both sessions of the study. Each session started by introducing the two experimental tasks and suggestion methods (only for session 2). In session 1, participants then practiced the two tasks. In session 2, they practiced the scenarios with and without the two suggestion types in each task. The suggestion timing was shortened to 1/3 of the original intervals during practice to ensure they saw a suggestion. They then started the experiment where they were asked to complete each task as fast and as accurately as possible, and were encouraged to use intelligent suggestions if they were correct. They were given breaks between blocks. After session 2, they completed a post-study questionnaire.

5.6 Results - Session 1

Before the baseline task completion time was computed, the data was pre-processed to remove outliers that deviated more than three standard deviations from the mean ($mean \pm 3std$). This lead to 9 trials (1.25%) being discarded for the dense target selection task and 19 trials (2.64%) being discarded for the text matching task. A total of 711 trails and 701 trials were left for analysis, respectively.

The completion times for both tasks followed log-normal distributions. Using the maximum-likelihood estimation, the calculated distribution parameters were $\mu = 1.13$, $\sigma = 0.25$ for the dense target selection task and $\mu = 1.88$, $\sigma = 0.60$ for the text matching task. Participants took an average of 3.21 seconds ($std = 0.86$) to complete the dense target selection task and an average of 7.77 seconds ($std = 4.6$)

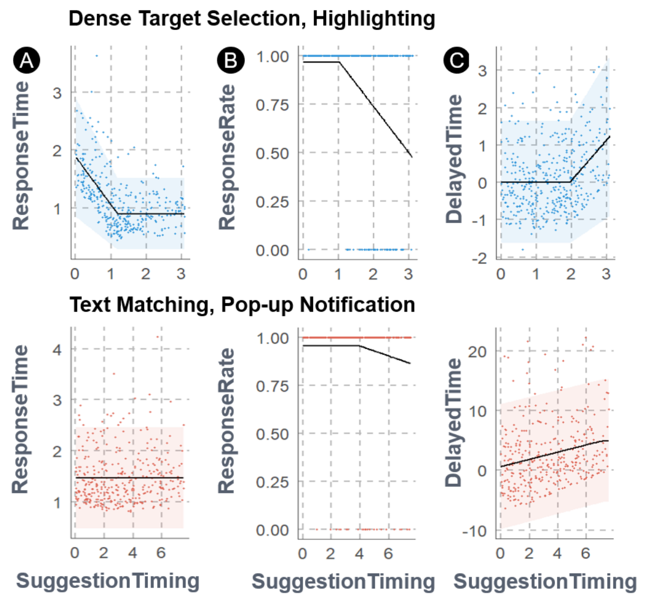


Figure 5: Examples of the modeling results for response time, response rate, and delayed time. The dots represent the data trials, the black lines are the model fitting results provided by MARS, and the ribbons indicate 95% CI.

to complete the text matching task. The overall accuracies were 94.09% and 100%, respectively.

5.7 Results - Session 2

Pre-processing the session 2 data involved first discarding trials where participants completed the task before an intelligent suggestion was displayed (i.e., 222 (7.71%) dense target selection trials and 447 (15.52%) text matching tasks). Additionally, trials outside $mean \pm 3std$, were also removed (i.e., 30 (1.04%) dense target selection trials and 40 (1.39%) text matching trials). This left 2628 trials and 2393 trials, respectively, for each task for analysis. The overall accuracy for the dense target selection task was 95.09% and 99.28% for the text matching task.

5.7.1 Response Time. Response time was defined the time elapsed between the appearance of a correct intelligent suggestion and a participant's selection of that suggestion. We used multivariate adaptive regression splines (MARS) to model the relationships between suggestion timing and response time. MARS was used because it tries to find multiple linear regression lines to fit data while balancing goodness-of-fit and simplicity. The linear regression lines were connected through hinge functions ($h(x-c) = max(0, x-c)$ or $h(c-x) = max(0, c-x)$ where c was a constant called knot) to provide non-linear approximations of the data. The maximum number of terms was set to two for the robustness of the model. The resulting equations for the four conditions are summarized in Table 1. Figure 5A shows graphic illustrations of the relationship between suggestion timing and response time of two example conditions .

5.7.2 Response Rate. Response rate was defined as the likelihood that participants accepted a correct suggestion. We applied MARS to model the relationship between the response rates and suggestion timings directly. Specifically, suggestion timing was used as a predictor and the accuracy of the suggestion was as the target variable (0: incorrect, 1: correct). The regression results then approximated the percentage of participants accepting a correct suggestion over time (Figure 5B). Table 1 summarizes the corresponding MARS models.

5.7.3 Delayed Time. Delayed time was the time delay that was incurred due to incorrect suggestions. For a given trial, it was infeasible to record the task completion time both with and without a suggestion (even if we repeated the trial, factors such as learning and familiarity would differ). Therefore, this metric was computed using the task completion time of each trial with an incorrect suggestion minus the average task completion time in the corresponding condition with no suggestion. The calculated distribution then allowed us to determine the average delay an incorrect suggestion would cause across different suggestion timings (Figure 5C). The delayed time data was fit into the MARS model for each condition. The results are summarized in Table 1.

5.8 Summary

Based on the data collection results, MARS models were able to simulate how participants would respond to an intelligent suggestion at different timings (Table 1). The models resulted in reasonable approximations of cost functions `Cost`(t) and benefit functions `Benefit`(t) for the two objectives. The gain of displaying an intelligent suggestion at timestamp t can thus be calculated via Equation 4 and 5. From the study results, it was also determined that the gain functions for the four conditions (Task Type × Suggestion Method) were quite different. Therefore, the four conditions were handled differently in later evaluations.

6 STUDY 2 - SIMULATION

The primary goal of the second study was to conduct a theoretical evaluation of the two suggestion timing optimization strategies - Optimal Thresholding (OT) and Reinforcement Learning (RL). To achieve this, a mock target prediction model that generated various data trials (D_{train}) during the two task scenarios was built. Simulations were run to estimate the gain of the optimization strategies. To constrain the search space, the study focused on applying highlighting suggestions for the dense target selection task, as it was less intrusive, and using pop-up notifications for the text matching task, as it led to quicker responses.

The following subsections first present the mock target prediction model that was used to generate D_{train} and then introduce the four simulation experiments that were undertaken. In Simulation 1, the performance of OT was bench-marked for the time saved for participants versus the suggestion usage percentage. The performance of the baselines that leveraged the design heuristics were also used to determine thresholds. In Simulation 2, RL was applied for optimization. In Simulation 3, multi-objective optimization (i.e., time saved and usage percentage) was run with OT.

Table 1: Summarization of the modeling results from MARS (multivariate adaptive regression splines).

Task Type	Suggestion Method	Response Time	Response Rate	Delayed Time
Dense Target Selection	Highlighting	$0.90+0.83\cdot h(1.19-x)$	$0.97-0.24\cdot h(x-1.02)$	$0.01+1.08\cdot h(x-1.96)$
Dense Target Selection	Pop-up Notification	$1.13+0.13\cdot h(x-1.60)$	$1.00-0.24\cdot h(x-0.98)$	$0.57+2.25\cdot h(x-2.52)$
Text Matching	Highlighting	2.91	0.90	$0.66+0.84\cdot h(x-1.29)$
Text Matching	Pop-up Notification	1.47	$0.96-0.03\cdot h(x-3.90)$	$4.94-0.61\cdot h(7.13-x)$

Table 2: Testing results when using Optimal Thresholding (OT) and Heuristic Thresholding (HT) regarding the dense target selection (DTS) task and the text matching (TM) task.

	Task	Strategy (Th.)	Time Saved/Usage%	% Improved
Time saved	DTS	OT (0.47)	0.4073s (0.3202s)	39.39%
	DTS	HT (0.85)	0.2922s (0.3645s)	-
	TM	OT (0.98)	1.6211s (1.7946s)	260.89%
	TM	HT (0.50)	0.4492s (1.1440s)	-
Usage %	DTS	OT (0.81)	65.69% (18.30%)	0.36%
	DTS	HT (0.85)	65.45% (20.42%)	-
	TM	OT (0.96)	87.17% (18.53%)	51.52%
	TM	HT (0.50)	57.53% (15.63%)	-

6.1 Target Prediction Model Mock-up

As most models' prediction accuracy values seem to follow sigmoid curves over task progression (e.g., [2, 15, 20, 36]), we simulated a similar model by mimicking the observed sigmoidal relation between accuracy and time to generate D_{train}. Specifically, for each trial, we first sampled trial length T based from the log-normal distribution found in the first session of Study 1 (Figure 6A). Then, a sigmoid function of task progression regarding prediction accuracy was computed (Figure 6B-C) and deviations (i.e., spikes and dips) were added to the sigmoid function (Figure 6D). More details of this mock-up target prediction model can be found in Appendix B.1.

The mock-up target prediction model was limited in that it only mimicked the appearance of the confidence curves, so it did not capture the inherent decision information of a real target prediction model. However, if the optimization strategies worked with a pseudorandom model, then they may also work with an actual target prediction model. Next, we present simulation results based on 30,000 trials generated by the mock-up prediction model for each task scenario. The trials were separated such that 90% were used for training and 10% were used for testing. Among the training data, 10% was used for hold-out validation. We present only testing results in the paper while readers can find the validation results in Appendix B.2.

6.2 Simulation 1: Optimal Thresholding

The Optimal Thresholding (OT) strategy sought to learn an optimized confidence threshold from the dataset that would lead to the best gain. To achieve this, different confidence values were tested ($q_t \in [0,1]$, 0.01 per step) and the corresponding gain was calculated using Equation 4 and 5 from the first study. Figure 7 presents two examples of how the gain in the time saved condition changed as the confidence threshold q_t varied. The optimized threshold was quite different for the dense target selection task ($thres = 0.47$) compared to the text matching task ($thres = 0.98$).

To benchmark the performance of OT, we picked a threshold that worked the best on the validation dataset and produced the corresponding results on the testing dataset. The baseline (i.e., Heuristic Thresholding) for the dense target selection task was determined to be $thres = 0.85$, which was directly appropriated from a similar point-and-select task in the literature with sigmoidal prediction curves [20]. The baseline for the text matching task was $thres = 0.50$, which was used to predict participant selections in a search-intensive task like our text matching task (i.e., users' intended ingredients in a sandwich-making task [36]).

From the results, the optimized threshold was found to save 0.1 seconds more than the baseline in the dense target selection task (around 40% of improvement) and 1 second more than the baseline in the text matching task (around 260% of improvement; Table 2). The optimized threshold also led to an 87% suggestion usage percentage in the text matching task (around 50% of improvement). The optimized thresholds were quite different for the dense target selection task for the time saving optimization ($thres$ = 0.47) and usage percentage optimization ($thres = 0.81$), while being similar for the text matching task (0.98 *vs.* 0.96).

6.3 Simulation 2: Reinforcement Learning

RL can potentially provide tailored solutions based on the target prediction confidence profile of each trial (e.g., the speed of increase of the model confidence value) by finding an appropriate threshold to display suggestions that works for that specific profile. To achieve this, model-free RL techniques were leveraged because there was a lack of transition dynamics models for our problem. Thus, the model-free RL agents observed the model confidence estimates q_t from a target prediction model trained on D_{train}, which were replayed multiple times to the agent. On each trial, the agent explored different action sequences (i.e., displayed an intelligent suggestion or not) to ultimately find the optimal action sequence for a given q_t trajectory that would lead to the maximum gain.

6.3.1 *Problem Formulation.* The key elements of the RL agents were:

- *Observation*: For a specific timestamp t, the agent received the following observation $\{p_1, p_2, ..., p_m, d_t\}$. The probability values $\{p_1, p_2, ..., p_m\}$ were the model confidence values produced by the target prediction model over time. The integer m was the memory size of the agent. The list acted like a first-in-first-out queue where p_m represented the most recent confidence value provided by the prediction model and p_1 represented the least recent. The float d_t recorded the last timestamp when a suggestion was displayed.

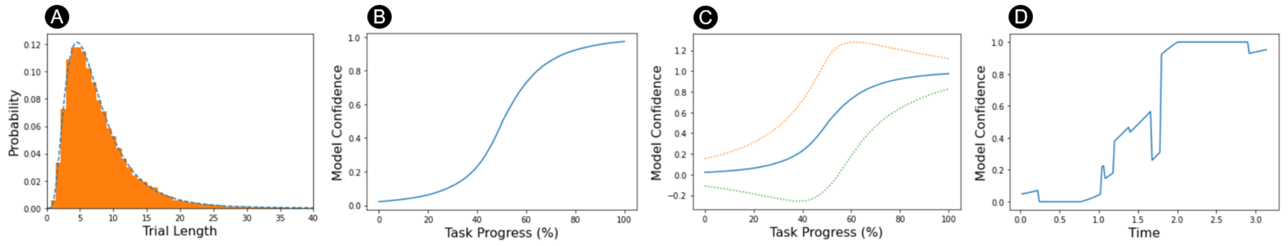


Figure 6: The trial generation process for the mock-up target prediction model. (A) The model first computes the trial length based on the log-normal distribution found in Study 1 for task completion time. (B) The model forms a sigmoid function of task progression with respect to prediction accuracy. (C) The sigmoid function varies within a predefined region (the dashed lines indicate the 95% CI). (D) The model adds deviations (i.e., spikes and dips) to the trial.

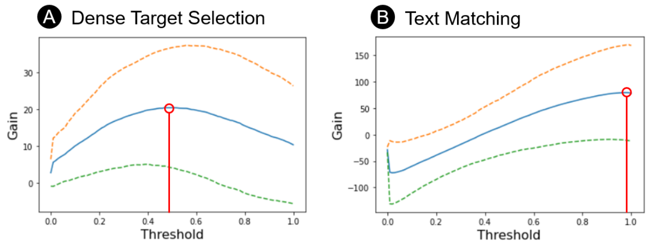


Figure 7: The expected gain for maximizing time savings for participants (y-axis) when using different confidence thresholds (x-axis) based on the validation dataset. The unit of gain is a timestamp, where the time saved in seconds equals 0.02 · timestamps. Dashed lines represent *mean*±*std*.

- *Action*: The agent could take the following two actions based on the observation $\{display, not\ display\}$. The *display* action represented displaying an intelligent suggestion, so d_t was updated to the current timestamp t. The *not display* action hid the suggestion.
- *Reward*: Three reward settings were used to train the RL agents. The first was r_1, where $r_1^t = \texttt{Gain}(t)$ if a suggestion was displayed at t, otherwise $r_1^t = 0$. The second reward setting, r_2, sought to solve the reward sparsity issue in r_1. Specifically, reward shaping was performed when the suggestion wasn't displayed: $r_2^t = \texttt{Gain}(t)$ if a suggestion was displayed at t, otherwise $r_2^t = -k \cdot p_m$. We used a hyper-parameter k to penalize the action of not displaying any suggestion. An agent received more penalties if it did not display a suggestion when the model confidence was high (p_m). The third reward r_3^t also leveraged the benefit of dense rewards, but removed the agents' reliance on the penalty factor k, which may have negative impacts on true reward maximization. Here, $r_3^t = \texttt{Gain}(t) - r_3^{t-1}$ (where $r_3^0 = 0$) at a timestamp t. This essentially rewarded the agent based on how good it performed on a particularly timestamp t, by computing the contribution of agent's action at t towards the gain. More details can be found in Appendix B.1.4.
- *Episode End Criteria*: The current episode ended if t was larger than the maximum length of the trial T, or d_t was larger than 0 (which meant a suggestion was displayed).
- *Initialization*: p_m was initialized to the first confidence value produced by the target prediction model, while all other probability values were set to 0. d_t was initialized to 0.

Table 3: Testing results of RL regarding regarding the dense target selection (DTS) task and the text matching (TM) task.

Task	Strategy	Time Saved	% Improved
DTS	PPO-MLP	0.4087s (0.3285s)	39.87%
DTS	ACER-LSTM	0.4084s (0.3362s)	39.77%
TM	PPO-MLP	1.6050s (1.7877s)	257.30%
TM	ACER-LSTM	1.5671s (1.7328s)	248.86%
Task	**Strategy**	**Usage%**	**% Improved**
TM	PPO-MLP	87.31% (18.05%)	51.76%

6.3.2 *Methodology.*

OpenAI Gym [16] and Stable Baselines [34, 56] were used to build and train the RL agents. Our experimentation demonstrated that PPO2 with MLP policies was a lightweight and effective solution and ACER with LSTM was powerful but may take longer to train. We thus used these two strategies for final benchmarking. Since training these RL agents consumes a lot of resources, for demonstration purposes, we only optimized agents for the time saving objective and one agent for the usage percentage objective. More training details can be found in Appendix B.1.5.

6.3.3 *Results.*

The results showed that RL agents could provide around 40% of improvement in the dense target selection task and 260% of improvement in the text matching task as compared to Heuristic Thresholding (Table 3). Compared to the results from Section 2, OT and RL led to very similar performance improvement in the two task scenarios; while RL did produce dynamic thresholds for each trial. We will return to this in later sections (Section 7.2.3 and 8.2).

6.4 Simulation 3: Multi-Objective Optimization

So far, our approach focused on optimizing a single objective e.g., time saved or usage percentage. However, designers may need to find optimal decisions in the presence of trade-offs between two or more conflicting objectives (e.g., minimizing task completion time and maximizing accuracy) in many applications. Multi-objective optimization is useful in such settings, when more than one objective function need to be optimized simultaneously. Therefore, we explored Pareto Frontier-based multi-objective optimization technique [45, 48], which generates a set of acceptable trade-off optimal

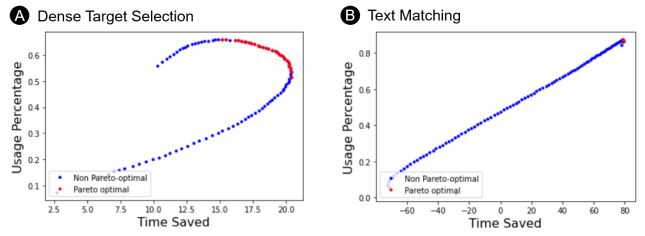


Figure 8: Optimizing both the time saved for participants and the suggestion usage percentage with Pareto Frontier-based multi-objective optimization.

solutions, to optimize the two objectives—time saved and suggestion usage percentage simultaneously.

A given condition is called Pareto optimal if one dimension (i.e., objective) could not be improved without worsening other dimensions (i.e., objectives). In our case, we computed the gain of timing saving and usage percentage for each condition q_t and plotted them on a two dimensional xy-plane (Figure 8). A Pareto optimal point was identified if there was no point on the plane that was better in both x and y dimensions. The corresponding threshold q_t of the point was then retraced.

Following the above method, we identified thresholds that could optimize both objectives simultaneously. Thirty-two Pareto optimal values were identified for the dense target selection task ($Thres = 0.47, 0.50 - 0.78, 0.80 - 0.81$) and three Pareto optimal values were identified for the text matching task ($Thres = 0.96 - 0.98$). The results indicated that the time saved and usage percentage objectives were somewhat conflicting in the pointing task but not in the text matching task. Thus COBO can help practitioners who want to trade-off various optimization objectives.

6.5 Summary

These simulation experiments demonstrated different facets of optimization strategies using COBO. The experiments showed how, theoretically, OT and RL were both effective at determining the optimal timings at which to show an intelligent suggestion, while the performance difference between the two strategies was small. We found that for the dense target selection task, an intelligent suggestion should be displayed when the model confidence reached 0.47 for optimizing time saved for users and 0.81 for optimizing suggestion usage percentage. For the text matching task, an intelligent suggestion should be displayed when the model confidence reached around 0.96-0.98 for optimizing both objectives.

It was also found that a non-optimized threshold could lead to much worse performance (e.g., 1 second longer in task completion time and a 30% smaller suggestion usage percentage in the text matching task) compared to an optimized strategy based on COBO. Not all intelligent suggestions were shown to be beneficial, however. Displaying suggestions early in the text matching task lead to a negative gain in terms of task completion time.

7 STUDY 3 - VALIDATION

The third study consisted of two empirical user experiments of COBO because of the high number of conditions. The first one compared the time saved and suggestion usage % for Optimal Thresholding (OT) and Heuristic Thresholding (HT), finding that OT saved participants more time and led to a higher suggestion usage percentage in the text matching task. The second experiment compared OT and RL strategies and found that OT and RL lead to similar performance.

7.1 Validation 1 - Optimal Thresholding vs. Heuristic Thresholding

The goal of validation experiment 1 was to empirically verify the effectiveness of Optimal Thresholding in comparison with Heuristic Thresholding. We also included a No Suggestion condition to help contextualize the impact of suggestion conditions relative to when the interface offers no suggestions.

7.1.1 Participants and Apparatus. Another 26 participants were recruited (i.e., fourteen women, eleven men, and one non-binary). Their ages ranged from 22 to 65 ($mean = 36.1, std = 12.8$). All participants had normal or corrected-to-normal vision and were right-handed. 23 participants had used VR devices 0-5 hours per week, two used 5-10 hours per week, and one had never used any VR device before. The same apparatus was used as in the first study.

7.1.2 Methodology. Participants experienced both task scenarios (i.e., dense target selection and text matching). There were four conditions (Strategy) for the dense target selection task: optimized thresholds for time saved (*TS*, $thres = 0.47$), optimized thresholds for suggestion usage percentage (*UP*, $thres = 0.81$, which was close to HT $thres = 0.85$ from a selection task [20]), balanced optimization for both objectives (*BA*, $thres = 0.64$), and no intelligent suggestions (*NS*). Similar to Study 2, we used highlighting suggestions for the dense target selection task.

There were three conditions (Strategy) for the text matching task: balanced optimization based on OT (*BA*, $thres = 0.97$), HT baseline (*HT*, $thres = 0.50$ from a search-heavy, mentally-demanding task [36]), and no intelligent suggestions (*NS*). The time saved ($thres = 0.98$), suggestion usage percentage ($thres = 0.96$), and balanced ($thres = 0.97$) optimization conditions were combined in this task as the thresholds were very close. We used pop-up notifications for the text matching task. This design enabled us to investigate multiple factors while keeping the study size reasonable at seven experimental conditions.

48 trials of predictions were generated for each task scenario using the mock-up target prediction model from Study 2. Each trial contained the probability of the model making a correct suggestion (i.e., model confidence) over a fixed period of time. The different thresholding strategies were then applied to each trial to decide the timing of showing a suggestion. The 48 trials were fixed across conditions to minimize the variances caused by the target prediction model. The average global centerline of the 48 trials followed a sigmoid curve. The final correctness of the prediction (i.e., a predicted candidate which participants visually perceived) was determined based on the confidence value when displaying a suggestion. For example, if a strategy decided to display the suggestion when the confidence value was 0.6, the final prediction then had 60% chance to be correct. Among the 48 trials, the first 3 trials were treated as practice trials. In total, 8190 trials were recorded (= 26 participants × 7 conditions × 45 repetitions) during this experiment.

A similar experimental procedure was employed as the first study. However, in this study, after completing each condition, participants were asked to complete a questionnaire that had three 7-point Likert scale questions probing easement, physical workload, and mental workload. The order of the task scenarios was randomized and the conditions within the scenarios were counterbalanced. The order of the formal trials were also randomized, however, the practice trials were always the same.

7.1.3 Analysis and Results. While data was initially collected for 26 participants, P1, P14, P19, and P26 were excluded as they never used intelligent suggestions in one or both of the tasks. The trials where participants had finished before the suggestion appeared (i.e., 169 (3.61%) dense target selection trials and 518 (14.76%) text matching tasks) were removed from the dataset. Because a mock-up target prediction model was used, there could have been trials where participants finished earlier than the pre-determined time period. Thus, only trials where an intelligent suggestion was displayed were considered. We also removed outliers ($mean \pm 3std$) (i.e., 45 (0.96%) dense target selection trials and 42 (1.20%) text matching tasks). These preprocessing steps resulted in 6156 trials remaining for analysis (i.e., 3746 trials for dense target selection and 2410 trials for text matching). The trials were later averaged across participant and condition. The overall accuracy was 93.14% for the dense target selection task and 98.98% for the text matching task.

For the dense target selection task, a linear mixed model with arcsinh transformation (as determined by the `bestNormalize` package) suggested that Strategy had a significant main effect on task completion time ($F = 5.02, p = .003$). A post-hoc analysis with Bonferroni correction showed that the completion time in *NS* was significantly longer than *BA* ($p = .002$), and marginally significant longer than *TS* ($p = .084$) and *UP* ($p = .135$) (all other $p > .887$). Another linear mixed model with exp transformation indicated that Strategy had a significant main effect on suggestion usage percentage ($F = 65.69, p < .001$). Post-hoc analysis suggested that usage percentages of *UP* ($p = 0.056$) and *BA* ($p = 0.140$) were marginally significant higher than *TS*. See Figure 9A-B for an overview.

For the text matching task, a linear mixed model with sqrt transformation suggested that Strategy had a significant main effect on task completion time ($F = 59.79, p < .001$). A post-hoc analysis showed that participants performed significantly faster in *BA* than *HT* ($p < .001$) and *NS* ($p < .001$). *HT* was also found to have a significantly shorter task completion time than *NS* ($p < .001$). Another linear mixed model with exp transformation suggested that Strategy had a significant main effect on suggestion usage percentage ($F = 420.45, p < .001$). A post-hoc analysis indicated that *BA* had a significantly higher suggestion usage percentage than *HT* ($p < .001$). See Figure 9C-D for an overview.

For the subjective questions, pair-wise comparisons (with Bonferroni correction) identified that *BA* led to lower mental workload ($p = .012$), and were possibly easier to use ($p = .053$), than *NS* in the text matching task. This suggests that using an intelligent suggestion could reduce workload and improve user experience.

7.1.4 Discussion. The empirical results demonstrated the effectiveness of the COBO optimization framework for the text matching task. As expected from the theoretical evaluation, the optimized condition (*BA*) led to shorter task completion times and higher suggestion usage % than the baseline conditions (*HT* and *NS*).

The benefits due to COBO were more obvious in the text matching task compared to the dense target selection, mainly because the dense task was very rapid and, as such, it was more difficult to have substantial differences in suggestion timings (thus their effect on time saved for users and suggestion usage percentage). However, the patterns across the two tasks were consistent. The significantly higher suggestion usage in text matching, in particular, could be impactful in lowering user's effort, which is suggested in the lower mental load scores of the balanced optimization.

7.2 Validation 2 - Optimal Thresholding vs. RL

The primary goal of the validation experiment 2 was to compare Optimal Thresholding (OT) vs. RL strategies for time saved and suggestion usage percentage. Based on the findings from validation 1, in this study, only the text matching task was used, as it was more likely to lead to verifiable performance differences in an empirical user study than the dense target selection task.

7.2.1 Participants and Apparatus. 12 participants (6 women, 5 men, and 1 non-binary) who had participated in the first validation study were recruited for the second validation study. Since the time interval between validation experiment 1 and 2 was more than a week and the strategy differences were hard to verify by seeing only the suggestion itself, it was presumed to be reasonable to reuse participants. Participants' age ranged from 22 to 63 ($mean = 35.9, std = 10.9$). The same apparatus were used as in validation study 1.

7.2.2 Methodology. The study employed a 2×2 within-subject design: Objective (time saved and suggestion usage percentage) × Strategy (OT and RL). Based on Study 2, $thres = 0.98$ was used for time saved optimization and $thres = 0.96$ was used for suggestion usage percentage optimization. The PPO-MLP agent from Study 2 was used.

The same 48 trials were used to generate the corresponding suggestion timing in each condition, and a similar study protocol was employed as validation study 1. In total, 2160 trials were collected (= 12 participants × 2 objectives × 2 strategies × 45 repetitions).

7.2.3 Analysis, Results, and Discussion. After removing outliers ($mean \pm 3std$, 11 trials, 0.51%) and trials where participants finished before the suggestion appeared (709 trials, 32.8%), 1440 trials remained for analysis. The overall accuracy was 99.59%.

A linear mixed model with sqrt transformation was not able to identify that Strategy had a significant main effect on task completion time ($F = 0.18, p = .674$). Another linear mixed model with Yeo-Johnson transformation was not able to identify that strategy had a significant main effect on suggestion usage percentage ($F = 0.74, p = .397$). Strategy was not shown to have significant main effects on any of the subjective scales. In summary, our results did not find any significant differences between OT and RL that lead to identifiable differences in the optimization metrics (Figure 10A-B).

We were further interested to see whether RL proposed different suggestion timings than OT in the 48 trials. For the time saved optimization, RL and OT led to a similar suggestion timing ($\Delta < 0.1$s) in most cases (72.9%). For 16.8% of the cases, the difference between them was > 0.5s. For usage percentage optimization, there were 68.8%

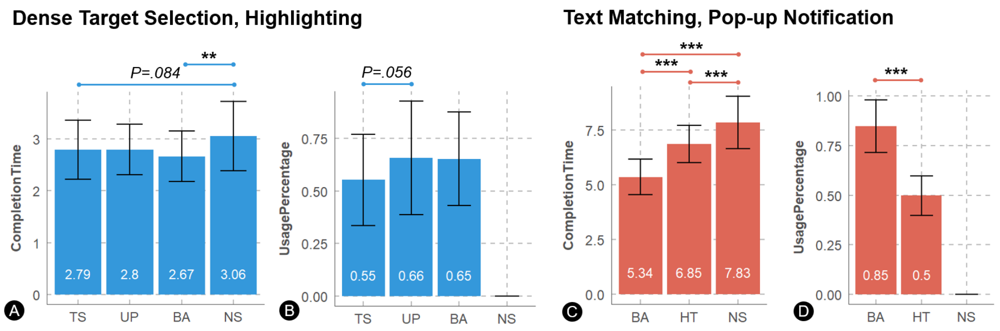


Figure 9: Results of average task completion time and suggestion usage percentage in the first validation experiment of Study 3. The four conditions in the dense target selection task were time saved optimization (TS), usage percentage optimization (UP), balanced optimization (BA), and no suggestion (NS). The three conditions in the text matching task were balanced optimization (BA), Heuristic Thresholding (HT), and no suggestion (NS). The error bars represent *mean*±*std*. ** means $p < .01$ and * means $p < .001$.**

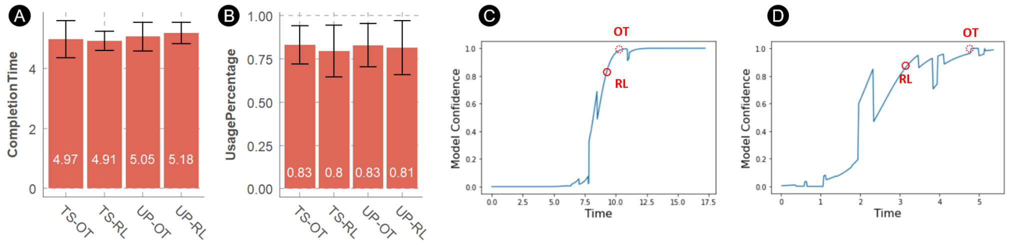


Figure 10: Results of average task completion time (A) and suggestion usage percentage (B) in the second validation experiment of Study 3. The four conditions were usage percentage optimization with RL (UP-RL) and Optimal Thresholding (UP-OT) and time saved optimization with RL (TS-RL) and Optimal Thresholding (TS-OT). The error bars represent *mean* ± *std*. (C) and (D) show example trials where RL and Optimal Thresholding (OT) yielded noticeably different suggestion timings. On average, RL saved 0.31s less in (C) and 1.79s more in (D) than OT.

trials where RL and OT led to a similar suggestion timing (Δ < 0.1s) and 8.3% trials that resulted in difference > 0.5s. In the trials with difference >0.5s, RL *always* attempted to display an earlier suggestion to save more time for users. On average, RL showed the suggestions 0.79s ($std.$=0.38s) earlier in these trials as compared to OT.

Figure 10C-D demonstrate two examples wherein RL finds different thresholds than OT. RL strategy seems to be observing the trend of the model confidence curve and displaying a suggestion once the curve is likely to plateau in the near future. Figure 10C shows a trial where RL saved 0.31 less than OT on average, and Figure 10D shows a trial where RL saved 1.79s more. Thus, RL is certainly able to learn a strategy that results in dynamic thresholds that match OT performance on average, but it remains to be seen if/when RL may be able to outperform optimal thresholds.

8 DISCUSSION

We've conducted a series of three studies that demonstrated the theoretical and empirical effectiveness of our COBO (cost-benefit optimization) framework for suggestion timing optimization. In this section, we further reflect on our experiences in terms of the cost and benefit quantification of the two optimization objectives and the strength of RL as an optimization strategy as compared to Optimal Thresholding. We also discuss the generalizability of the framework to other applications and the limitations of our studies.

8.1 Optimization Objectives

Our work demonstrates a successful optimization of two objectives: time saved by users and suggestion usage percentage. The COBO framework is designed to help optimize various objectives, either individually or simultaneously, as long as a cost and benefit quantification method can be determined. We used data collected from participants (Study 1) to construct cost and benefit functions with variables such as response times, response rates, and delayed times. The validation studies indicated that the constructed cost and benefit functions were good approximations of the ground truth.

The time savings in our case, even though significant, are small especially in the dense target selection task. However, existing work has shown that users prefer intelligent suggestions despite negative time costs [54] because they were considered less physically demanding and effortful. The fact motivated us to quantify the benefit of intelligent suggestions beyond performance improvements. While usage percentage is an effective proxy that assumes that higher suggestion usage is always beneficial for a user to lower their interaction friction [38], a highly promising avenue for future work is in optimizing directly for effort, physical and mental-demand especially as we become better at real-time estimations of quantities like arm fatigue [18] and satisfaction [24, 53].

For simplicity, we omitted some rare conditions during cost-benefit quantification. For example, we excluded the trials where users mistakenly triggered the selection of an incorrect suggestion. Such instances were very uncommon (0.4% overall) and did not significantly impact the suggestion usage percentage or the time cost of an incorrect suggestion. However, future endeavors can extend our framework to consider mistaken triggering of an incorrect suggestion especially if those instances are not rare and/or if they require a costly recovery from the mistake [12, 40]. One simple way might be to consider modeling this as a constant time cost (e.g., recovery time).

8.2 RL as an Optimization Strategy

We found that RL was able to learn a successful strategy and produce dynamic thresholds across trials. However, RL's dynamic thresholds weren't able to outperform the single optimal threshold on average in our simulation and validation study.

As we report, there were a small, but significant percentage of trials where RL's suggestion timing differed by >0.5s compared to OT. However, we did not find any big discernible patterns in these trials compared to others. It will be worth investigating task contexts where the percentage of such trials is higher. Another reason for RL's similar performance to OT might be that the room of improvement for RL was small, as Optimal Thresholding (OT) already performed very well. The analysis demonstrated that even the theoretical maximum of a perfect agent (i.e., agent that maximizes the gain by knowing the whole trial profile) can lead to no larger than 0.18s and a 4.3% improvement over OT in task completion time and suggestion usage percentage, respectively, with our dataset. It will be interesting to see if there are contexts where OT does not achieve performance close to the theoretical maximum.

We can propose two variables to explore here that may help diversify our task context. First, is to look at trials with durations that are much more variable. Looking at the validation study data more closely, we found a weak correlation between the time-saving differences ($RL - OT$) and trial length ($R^2 = 0.10$) which indicated that the RL agent saved more time than OT in longer duration trials. Second, is to look at target prediction models that are not sigmoidal in nature (as an example, models that start with a high prior confidence using earlier user activity), and may follow patterns that cannot be easily captured using a single OT.

RL may also prove to be useful in scenarios where an interface wants to show more than one intelligent suggestion and the suggestions get updated based on users' behavior. It might be hard to directly apply OT in these scenarios. Also, in case a designer wants to enable different suggestion types within the same task (example, both highlighting and pop-up notification), an RL agent could choose the most appropriate suggestion type based on the gain of those options at different timings. An interesting area of exploration is the long-term use of such intelligent suggestion interfaces. A user may form an expectation of how well the model performs, which can in turn influence their response behavior, thus changing the cost-benefit quantification over time. An online RL agent may also prove useful in such cases.

8.3 Applications

This research has demonstrated the application of COBO in two task scenarios (dense target selection and text matching) and two objectives (minimizing user task completion time and maximizing intelligent suggestion usage). The two tasks and suggestion types were intentionally chosen to be representative of popular use cases. The dense target selection task aims to simulate physically-demanding tasks where users need to select objects in cluttered environment [46, 64], and the text matching task mimics real-world search-heavy scenarios such as searching for ingredients from a receipt [25, 63]. Object highlighting and pop-up notification are both common visualizations to inform users about system events [57]. Additionally, in Appendix B.3, we also present results on successfully applying COBO on a dataset from the literature which records hand movement trajectories when reaching virtual objects at different locations. We further envision COBO being extensible to other tasks and facilitation.

8.3.1 Extending to other tasks. The framework can be retrained for other applications that want to leverage intelligent predictions using target prediction models that rely on hand, head, gaze, and other contextual information [31, 70] in selection tasks such as pointing, visual search, and text-entry. By following the COBO framework, practitioners may choose different models, objectives, and cost-benefit quantification methods which are tailored for their applications. Overall, based on our user-centric computational framework, designers are more likely to provide intelligent suggestions that support their intended goals, rather than leading to unexpected outcomes [50, 54].

8.3.2 Extending to other facilitation. COBO's framework can also be extended to facilitate techniques other than intelligent suggestions such as expanding [44] or auto-selecting [1, 4] a predicted target, as well as for more than one suggestions simultaneously or sequentially.

8.4 Target Prediction Model

The current research builds on certain assumptions to simplify the complex problem space. One assumption is the use of a mock-up target prediction model, as we wanted to simulate a highly representative prediction model, rather than choosing one at random. Therefore, we carried out a literature survey to extract the commonalities among prediction models and then created a simulation from those commonalities (Section 6.1). However, our inspirations were from human behavior models of target reaching [20] and searching [36] where the model prediction accuracy was typically high during the later stage of the task because the selection indicator (e.g., hand or gaze point) was "approaching" or "almost on" the target and the user was just "fine-tuning" the selection of the target. For example, in the text matching task, we imagined that the gaze direction would reach the targeted object way before the controller-based manual pointing selection (i.e., the model has very high confidence based on gaze features no matter the position of the hand pointer), as Huang et al. [36] could correctly anticipate the intended object through gaze sequences 1.8s before a speech request. We acknowledge that there are other types of models that may not have such rich features. Future work can deploy this framework to any prediction model to test it on new use cases. This, however, did mean that the intelligent suggestions were not delivered dynamically based on a user's behaviour. For experimental control, it was important that this be the case while developing and validating the COBO framework. However, future research should investigate how the framework responds to a real prediction model.

One additional consideration of the current approach is that it requires a dataset of model confidence curves to calculate user-centric costs and benefits over time. In a real scenario where a designer has a target prediction model and its training dataset, the training dataset should contain trials with necessary features (e.g., user behavior data, completion times) so the designer can directly use those for confidence curve generation and cost-benefit computation (see Appendix B.3 for an example). In a condition where the feature dataset is missing, another possible solution is to apply models to simulate user behavior. During the planning phase of this research, our initial idea was to use existing computational models (e.g., minimum jerk model) to generate a large volume of user behavior data. However, we encountered two challenges. First, we did not know how users would behave according to correct/incorrect suggestions that appeared at different timings (so it was hard to incorporate this element into the model). Second, a user behavioral model for the text matching scenario is still largely underexplored (unlike bio-mechanical behavior modeling for pointing and reaching as in Cheema et al. [18] and Fischer et al. [23]). Therefore, we decided to collect new data from real users. However, we do believe using model-generated datasets for user cost-benefit quantification can be helpful in the future with more advances in the field.

9 CONCLUSION

Predictive systems are helpful ways to lower input friction and improve user experiences in current VR/AR systems [38]. Specifically, selection facilitation techniques that leverage target prediction models can alleviate the need for manual pointing and visual search, and can potentially lead to quicker, easier, and more comfortable interaction. While current target prediction models only offer *which* target a user intends to select, we built a framework (COBO) that helps determine *when* an intelligent suggestion should be displayed to maximize its benefits.

COBO is a computational framework that determines the optimal timing of an intelligent suggestion for each interaction based on user-centric costs and benefits. In a set of studies, we demonstrated that COBO is effective at determining the optimal timing of intelligent suggestions. The first study focused on measuring and quantifying the costs and benefits of an intelligent suggestion displayed at different timings when trying to satisfy two objectives (i.e., time saved for users and suggestion usage percentage) during two tasks (i.e., dense target selection and text matching). We then run simulations with two optimization strategies (i.e., Optimal Thresholding and RL) for single- and multi-objective optimizations. We found both Optimal Thresholding and RL led to better performance compared to heuristic-based thresholding approaches. For example, both optimization strategies led to around 40% improvement in terms of task completion time in the dense target selection task and 260% improvement in the text matching task. We also demonstrated the effectiveness of COBO for multi-objective optimization. The third study contained two validation experiments that compared Optimal Thresholding, RL, heuristic-based thresholding, and no suggestion conditions. The experimental results suggested that COBO-based optimization strategies led to shorter task completion times and higher suggestion usage percentages, and were preferred by participants in the text matching task when compared to baselines.

From both theoretical and empirical perspectives, we showed that an optimized strategy based on COBO can perform significantly better than non-optimized heuristic-based approaches in maximizing the time saved by users and increasing suggestion usage percentages. Overall, we envision the introduced framework will unlock effective intelligent suggestions, which will benefit future predictive systems.

ACKNOWLEDGMENTS

We thank Ben Lafreniere, Kashyap Todi, and many others from Meta Reality Labs Research for insightful discussions. We also thank Michael Frederick and other team members for their help with user studies. Icons in figures are from Flaticon.com and 105 Colorful 2D Planet Icons in Unity Asset Store.

REFERENCES

[1] Bashar I. Ahmad, Patrick M. Langdon, Simon J. Godsill, Richard Donkor, Rebecca Wilde, and Lee Skrypchuk. 2016. You Do Not Have to Touch to Select: A Study on Predictive In-Car Touchscreen with Mid-Air Selection. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (Ann Arbor, MI, USA) *(Automotive'UI 16)*. Association for Computing Machinery, New York, NY, USA, 113–120. https://doi.org/10.1145/3003715.3005461

[2] Bashar I. Ahmad, Patrick M. Langdon, Simon J. Godsill, Robert Hardy, Eduardo Dias, and Lee Skrypchuk. 2014. Interactive Displays in Vehicles: Improving Usability with a Pointing Gesture Tracker and Bayesian Intent Predictors. In *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (Seattle, WA, USA) *(AutomotiveUI '14)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/2667317.2667413

[3] Bashar I Ahmad, James Kevin Murphy, Simon Godsill, Patrick M Langdon, and Robery Hardy. 2017. Intelligent interactive displays in vehicles with intent prediction: A Bayesian framework. *IEEE Signal Processing Magazine* 34, 2 (2017), 82–94. https://doi.org/10.1109/MSP.2016.2638699

[4] Bashar I Ahmad, James K Murphy, Patrick M Langdon, Simon J Godsill, Robert Hardy, and Lee Skrypchuk. 2015. Intent inference for hand pointing gesture-based interactions in vehicles. *IEEE transactions on cybernetics* 46, 4 (2015), 878–889. https://doi.org/10.1109/TCYB.2015.2417053

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[6] Ferran Argelaguet and Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers & Graphics* 37, 3 (2013), 121–136. https://doi.org/10.1016/j.cag.2012.12.003

[7] Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George Fitzmaurice. 2017. Experimental Evaluation of Sketching on Surfaces in VR. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5643–5654. https://doi.org/10.1145/3025453.3025474

[8] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* (2017).

[9] Christian Arzate Cruz and Takeo Igarashi. 2020. A Survey on Interactive Reinforcement Learning: Design Principles and Open Challenges. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference* (Eindhoven, Netherlands) *(DIS '20)*. Association for Computing Machinery, New York, NY, USA, 1195–1209. https://doi.org/10.1145/3357236.3395525

[10] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. 2005. Predictive Interaction Using the Delphian Desktop. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Seattle, WA, USA) *(UIST '05)*. Association for Computing Machinery, New York, NY, USA, 133–141. https://doi.org/10.1145/1095034.1095058

[11] Marc Baloup, Thomas Pietrzak, and Géry Casiez. 2019. *RayCursor: A 3D Pointing Facilitation Technique Based on Raycasting.* Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300331

[12] Nikola Banovic, Tovi Grossman, and George Fitzmaurice. 2013. The Effect of Time-Based Cost of Error in Target-Directed Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France)

(CHI '13). Association for Computing Machinery, New York, NY, USA, 1373–1382. https://doi.org/10.1145/2470654.2466181

[13] Xiaojun Bi and Shumin Zhai. 2013. Bayesian Touch: A Statistical Criterion of Target Selection with Finger Touch. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 51–60. https://doi.org/10.1145/2501988.2502058

[14] Pradipta Biswas, Gokcen Aslan Aydemir, Pat Langdon, and Simon Godsill. 2013. Intent recognition using neural networks and Kalman filters. In *International Workshop on Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*. Springer, 112–123. https://doi.org/10.1007/978-3-642-39146-0_11

[15] Ali Borji, Andreas Lennartz, and Marc Pomplun. 2015. What do eyes reveal about the mind?: Algorithmic inference of search targets from fixations. *Neurocomputing* 149 (2015), 788–799. https://doi.org/10.1016/j.neucom.2014.07.055

[16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[17] Tom Chandler, Maxime Cordeil, Tobias Czauderna, Tim Dwyer, Jaroslaw Glowacki, Cagatay Goncu, Matthias Klapperstueck, Karsten Klein, Kim Marriott, Falk Schreiber, et al. 2015. Immersive analytics. In *2015 Big Data Visual Analytics (BDVA)*. IEEE, 1–8. https://doi.org/10.1109/BDVA.2015.7314296

[18] 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* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376701

[19] Lung-Pan Cheng, Eyal Ofek, Christian Holz, Hrvoje Benko, and Andrew D. Wilson. 2017. Sparse Haptic Proxy: Touch Feedback in Virtual Environments Using a General Passive Prop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3718–3728. https://doi.org/10.1145/3025453.3025753

[20] Aldrich Clarence, Jarrod Knibbe, Maxime Cordeil, and Michael Wybrow. 2021. Unscripted Retargeting: Reach Prediction for Haptic Retargeting in Virtual Reality. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 150–159. https://doi.org/10.1109/VR50410.2021.00036

[21] Brendan David-John, Candace Peacock, Ting Zhang, T. Scott Murdison, Hrvoje Benko, and Tanya R. Jonker. 2021. Towards Gaze-Based Prediction of the Intent to Interact in Virtual Reality. In *ACM Symposium on Eye Tracking Research and Applications* (Virtual Event, Germany) *(ETRA '21 Short Papers)*. Association for Computing Machinery, New York, NY, USA, Article 2, 7 pages. https://doi.org/10.1145/3448018.3458008

[22] João Marcelo Evangelista Belo, Anna Maria Feit, Tiare Feuchtner, and Kaj Grønbæk. 2021. *XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445349

[23] Florian Fischer, Miroslav Bachinski, Markus Klar, Arthur Fleig, and Jörg Müller. 2021. Reinforcement learning control of a biomechanical model of the upper extremity. *Scientific Reports* 11, 1 (2021), 1–15. https://doi.org/10.1038/s41598-021-93760-1

[24] Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk. 2000. Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands) *(CHI '00)*. Association for Computing Machinery, New York, NY, USA, 345–352. https://doi.org/10.1145/332040.332455

[25] Christoph Gebhardt, Brian Hecox, Bas van Opheusden, Daniel Wigdor, James Hillis, Otmar Hilliges, and Hrvoje Benko. 2019. Learning Cooperative Personalized Policies from Gaze Data. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 197–208. https://doi.org/10.1145/3332165.3347933

[26] Eric J. Gonzalez, Parastoo Abtahi, and Sean Follmer. 2020. REACH+: Extending the Reachability of Encountered-Type Haptics Devices through Dynamic Redirection in VR. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 236–248. https://doi.org/10.1145/3379337.3415870

[27] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (San Francisco, California, USA) *(IUI '02)*. Association for Computing Machinery, New York, NY, USA, 194–195. https://doi.org/10.1145/502716.502753

[28] Tovi Grossman and Ravin Balakrishnan. 2005. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Portland, Oregon, USA) *(CHI '05)*. Association for Computing Machinery, New York, NY, USA, 281–290. https://doi.org/10.1145/1054972.1055012

[29] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*. PMLR, 1321–1330.

[30] Rorik Henrikson, Daniel Clarke, Thomas White, Frances Lai, Michael Glueck, Stephanie Santosa, Daniel Wigdor, Tovi Grossman, Sean Trowbridge, and Hrvoje Benko. 2020. Head-Coupled Kinematic Template Matching for Target Selection in Hangry Piggos. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3334480.3383176

[31] Rorik Henrikson, Tovi Grossman, Sean Trowbridge, Daniel Wigdor, and Hrvoje Benko. 2020. *Head-Coupled Kinematic Template Matching: A Prediction Model for Ray Pointing in VR.* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376489

[32] Niels Henze, Markus Funk, and Alireza Sahami Shirazi. 2016. Software-Reduced Touchscreen Latency. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Florence, Italy) *(MobileHCI '16)*. Association for Computing Machinery, New York, NY, USA, 434–441. https://doi.org/10.1145/2935334.2935381

[33] Lorenz Hetzel, John Dudley, Anna Maria Feit, and Per Ola Kristensson. 2021. Complex Interaction as Emergent Behaviour: Simulating Mid-Air Virtual Keyboard Typing using Reinforcement Learning. *IEEE Transactions on Visualization and Computer Graphics* 27, 11 (2021), 4140–4149. https://doi.org/10.1109/TVCG.2021.3106494

[34] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. 2018. Stable Baselines. https://github.com/hill-a/stable-baselines.

[35] Zhiming Hu, Andreas Bulling, Sheng Li, and Guoping Wang. 2021. Fixationnet: Forecasting eye fixations in task-oriented virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 27, 5 (2021), 2681–2690. https://doi.org/10.1109/TVCG.2021.3067779

[36] Chien-Ming Huang, Sean Andrist, Allison Sauppé, and Bilge Mutlu. 2015. Using gaze patterns to predict task intent in collaboration. *Frontiers in psychology* 6 (2015), 1049. https://doi.org/10.3389/fpsyg.2015.01049

[37] Chien-Ming Huang and Bilge Mutlu. 2016. Anticipatory robot control for efficient human-robot collaboration. In *2016 11th ACM/IEEE international conference on human-robot interaction (HRI)*. IEEE, 83–90. https://doi.org/10.1109/HRI.2016.7451737

[38] Tanya R Jonker, Ruta Desai, Kevin Carlberg, James Hillis, Sean Keller, and Hrvoje Benko. 2020. The Role of AI in Mixed and Augmented Reality Interactions. In *CHI2020 ai4hci Workshop Proceedings. ACM.*

[39] Fatemeh Koochaki and Laleh Najafizadeh. 2018. Predicting intention through eye gaze patterns. In *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 1–4. https://doi.org/10.1109/BIOCAS.2018.8584665

[40] Ben Lafreniere, Tanya R. Jonker, Stephanie Santosa, Mark Parent, Michael Glueck, Tovi Grossman, Hrvoje Benko, and Daniel Wigdor. 2021. False Positives vs. False Negatives: The Effects of Recovery Time and Cognitive Costs on Input Error Preference.. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3472749.3474735

[41] Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. 2007. Endpoint Prediction Using Motion Kinematics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 637–646. https://doi.org/10.1145/1240624.1240724

[42] Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. 2017. *3D user interfaces: theory and practice.* Addison-Wesley Professional.

[43] Huy Viet Le, Valentin Schwind, Philipp Göttlich, and Niels Henze. 2017. PredicTouch: A System to Reduce Touchscreen Latency Using Neural Networks and Inertial Measurement Units. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 230–239. https://doi.org/10.1145/3132272.3134138

[44] Michael McGuffin and Ravin Balakrishnan. 2002. Acquisition of Expanding Targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI '02)*. Association for Computing Machinery, New York, NY, USA, 57–64. https://doi.org/10.1145/503376.503388

[45] Kaisa Miettinen. 2012. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media.

[46] Martez E. Mott and Jacob O. Wobbrock. 2014. Beating the Bubble: Using Kinematic Triggering in the Bubble Lens for Acquiring Small, Dense Targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 733–742. https://doi.org/10.1145/2556288.2557410

[47] Atsuo Murata. 1998. Improvement of pointing time by predicting targets in pointing with a PC mouse. *International Journal of Human-Computer Interaction* 10, 1 (1998), 23–32. https://doi.org/10.1207/s15327590ijhc1001_2

[48] Patrick Ngatchou, Anahita Zarei, and A El-Sharkawi. 2005. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*. IEEE, 84–91. https://doi.org/10.1109/ISAP.2005.1599245

[49] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530* (2019).

[50] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. 2019. How Do People Type on Mobile Devices? Observations from a Study with 37,000 Volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services* (Taipei, Taiwan) *(MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, Article 9, 12 pages. https://doi.org/10.1145/3338286.3340120

[51] Phillip T. Pasqual and Jacob O. Wobbrock. 2014. Mouse Pointing Endpoint Prediction Using Kinematic Template Matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 743–752. https://doi.org/10.1145/2556288.2557406

[52] Philip Quinn and Andy Cockburn. 2008. The effects of menu parallelism on visual search and selection. In *Proceedings of the ninth conference on Australasian user interface-Volume 76*. 79–84.

[53] Philip Quinn and Andy Cockburn. 2020. Loss Aversion and Preferences in Interaction. *Human–Computer Interaction* 35, 2 (2020), 143–190. https://doi.org/10.1080/07370024.2018.1433040

[54] Philip Quinn and Shumin Zhai. 2016. A Cost-Benefit Study of Text Entry Suggestion Interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 83–88. https://doi.org/10.1145/2858036.2858305

[55] Antonin Raffin. 2018. RL Baselines Zoo. https://github.com/araffin/rl-baselines-zoo.

[56] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. 2019. Stable Baselines3. https://github.com/DLR-RM/stable-baselines3.

[57] Rufat Rzayev, Sven Mayer, Christian Krauter, and Niels Henze. 2019. Notification in VR: The Effect of Notification Placement, Task and Environment. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (Barcelona, Spain) *(CHI PLAY '19)*. Association for Computing Machinery, New York, NY, USA, 199–211. https://doi.org/10.1145/3311350.3347190

[58] Hosnieh Sattar, Mario Fritz, and Andreas Bulling. 2020. Deep gaze pooling: Inferring and visually decoding search intents from human gaze fixations. *Neurocomputing* 387 (2020), 369–382. https://doi.org/10.1016/j.neucom.2020.01.028

[59] Hosnieh Sattar, Sabine Muller, Mario Fritz, and Andreas Bulling. 2015. Prediction of search targets from fixations in open-world settings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 981–990. https://doi.org/10.1109/CVPR.2015.7298700

[60] Jonas Schjerlund, Kasper Hornbæk, and Joanna Bergström. 2021. *Ninja Hands: Using Many Hands to Improve Target Selection in VR*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445759

[61] Ronal Singh, Tim Miller, Joshua Newn, Liz Sonenberg, Eduardo Velloso, and Frank Vetere. 2018. Combining Planning with Gaze for Online Human Intention Recognition. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 488–496.

[62] Kashyap Todi, Gilles Bailly, Luis Leiva, and Antti Oulasvirta. 2021. *Adapting User Interfaces with Model-Based Reinforcement Learning*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445497

[63] Christina Trepkowski, David Eibich, Jens Maiero, Alexander Marquardt, Ernst Kruijff, and Steven Feiner. 2019. The effect of narrow field of view and information density on visual search performance in augmented reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 575–584. https://doi.org/10.1109/VR.2019.8798312

[64] Lode Vanacken, Tovi Grossman, and Karin Coninx. 2007. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. In *2007 IEEE symposium on 3D user interfaces*. IEEE. https://doi.org/10.1109/3DUI.2007.340783

[65] Datong Wei, Chaofan Yang, Xiaolong (Luke) Zhang, and Xiaoru Yuan. 2021. *Predicting Mouse Click Position Using Long Short-Term Memory Model Trained by Joint Loss Function*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411763.3451651

[66] Ryen W. White, Peter Bailey, and Liwei Chen. 2009. Predicting User Interests from Contextual Information. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 363–370. https://doi.org/10.1145/1571941.1572005

[67] Ryen W. White, Paul N. Bennett, and Susan T. Dumais. 2010. Predicting Short-Term Interests Using Activity-Based Search Context. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (Toronto, ON, Canada) *(CIKM '10)*. Association for Computing Machinery, New York, NY, USA, 1009–1018. https://doi.org/10.1145/1871437.1871565

[68] Haijun Xia, Ricardo Jota, Benjamin McCanny, Zhe Yu, Clifton Forlines, Karan Singh, and Daniel Wigdor. 2014. Zero-Latency Tapping: Using Hover Information to Predict Touch Locations and Eliminate Touchdown Latency. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 205–214. https://doi.org/10.1145/2642918.2647348

[69] Difeng Yu, Hai-Ning Liang, Kaixuan Fan, Heng Zhang, Charles Fleming, and Konstantinos Papangelis. 2019. Design and evaluation of visualization techniques of off-screen and occluded targets in virtual reality environments. *IEEE transactions on visualization and computer graphics* 26, 9 (2019), 2762–2774. https://doi.org/10.1109/TVCG.2019.2905580

[70] Difeng Yu, Hai-Ning Liang, Xueshi Lu, Kaixuan Fan, and Barrett Ens. 2019. Modeling Endpoint Distribution of Pointing Selection Tasks in Virtual Reality Environments. *ACM Trans. Graph.* 38, 6, Article 218 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356544

[71] Difeng Yu, Qiushi Zhou, Joshua Newn, Tilman Dingler, Eduardo Velloso, and Jorge Goncalves. 2020. Fully-occluded target selection in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 26, 12 (2020), 3402–3413. https://doi.org/10.1109/TVCG.2020.3023606

[72] Gregory Zelinsky, Zhibo Yang, Lihan Huang, Yupei Chen, Seoyoung Ahn, Zijun Wei, Hossein Adeli, Dimitris Samaras, and Minh Hoai. 2019. Benchmarking gaze prediction for categorical visual search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0. https://doi.org/10.1109/CVPRW.2019.00111

[73] Shumin Zhai, Carlos Morimoto, and Steven Ihde. 1999. Manual and Gaze Input Cascaded (MAGIC) Pointing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania, USA) *(CHI '99)*. Association for Computing Machinery, New York, NY, USA, 246–253. https://doi.org/10.1145/302979.303053

[74] Brian Ziebart, Anind Dey, and J. Andrew Bagnell. 2012. Probabilistic Pointing Target Prediction via Inverse Optimal Control. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces* (Lisbon, Portugal) *(IUI '12)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2166966.2166968

A STUDY 1 - DATA COLLECTION

A.1 Task Scenarios

Two task scenarios, representative of common interaction tasks that are effortful to perform, were employed. The dense target selection task represented a manually-intensive task, where participants needed to select a small object located at the center of a cluster [46, 64]. The text matching task served as a mentally-demanding task, where participants needed to find and select an object with text that matched a target text. This task simulated real-world, search-heavy scenarios like searching for ingredients from a receipt, finding street names on a map, or browsing through a menu [52].

A.1.1 Dense Target Selection Task. This task was inspired by existing literature on small and dense target selection [46, 64]. The goal was to select the earth icon at the center of a planet cluster (Figure 3, left). The cluster was surrounded by other planet icons, which were randomly sized and distributed to add noise to the task environment. This setting required participants to aim precisely [64] and simulated scenarios where participants need to select objects in a cluttered virtual scene (e.g., select a keychain in a messy room).

The angular size of the target was set to 1°, which was determined by previous research to be sufficiently challenging [70]. The angular distance, or required movement amplitude, was fixed to 90°, and the target was generated in a predefined list of locations that were no more than 30° away from the horizontal plane. This target placement required participants to rotate their heads to find the out-of-view object, which added physical workload, without requiring that they overextend their neck. The distractors that were located directly adjacent to the target were the same size as the target, while others were randomly sized between 0.6° and 2°.

Participants started the task by pointing at a button at a fixed center position. A blue 3D arrow then appeared to indicate the location of the target. The arrow was designed to minimize search time in this task [69]. Participants then followed the direction of the arrow to point at the target through the right-hand controller and pressed the trigger to confirm their selection.

A.1.2 Text Matching Task. This task was designed to require participants to perform a difficult visual search (mentally-demanding) [25, 63]. Participants were required to find a target text string that matched a prompt (Figure 3, right) in a 6×7 grid of texts strings.

The angular distance between the candidates was 10° horizontally and 2.8° vertically to make sure all objects were located within field of view of participants to minimize their physical workload (e.g., turning their bodies to search for the target). The object radius was set to 1.5° and all objects were placed on a spherical plane.

Participants started the task by memorizing the target string and selecting a button at a fixed center position. All candidate strings then appeared with the goal text reminder at the top of the grid. To complete a task trial, participants pointed at the target icon using the controller and pressed the trigger to select it.

A.2 Suggestion Method

Two suggestion methods were used in the study—a highlighting suggestion and a pop-up suggestion. With the highlighting suggestion, a blinking fluorescent outline was displayed around the suggested object (Figure 4 left). A symbol of Button A also appeared at a predetermined, unoccluded position close to the indicated object to depict that the object could be selected by pressing the Button A on the Touch controller. Participants could also cancel the suggestion by tilting the joystick to the right. Note that the highlighting suggestion was in-situ, so it remained at the object location without following the direction participants were looking.

With the pop-up notification suggestion, a suggestion window appeared at the bottom of the participant's current viewing direction (Figure 4 right) [57]. The suggestion presented either a predicted icon in the dense target selection task or a text string in the text matching task. When participants rotated their viewing direction, the pop-up notification followed the viewing direction using horizontal linear interpolation. Linear interpolation was not applied in the vertical dimension to avoid the suggestion being "stuck" on the head-mounted display, which may have caused visual discomfort. Like the highlighting suggestion, participants could quickly access the suggested object via the Button A or discard the suggestion by tilting the joystick to the right.

A.3 Example Data Trials

We show example data trials collected in session 2 in Figure 11.

A.4 Results - Session 2

Figure 12 shows the average response times and delayed times for the suggestion methods and task types. We performed significance tests with linear mixed models on response time and delayed time.

A.4.1 Response Time. Response time was defined the time elapsed between the appearance of a correct intelligent suggestion and a participant's selection of that suggestion. First, the Yeo-Johnson transformation, as chosen by the `bestNormalize` package in `R`, was applied to normalize the data. A linear mixed model was then used to identify whether different task types and suggestion methods lead to different response times across various suggestion timings. We set Task Type, Suggestion Method, and Suggestion Timing as fixed factors and Participant as a random factor. The linear mixed model indicated that there were interaction effects between Suggestion Method × Suggestion Timing ($F = 125.18, p < .001$) and Task Type × Suggestion Timing ($F = 49.47, p < .001$). As Task Type and Suggestion Method led to different response times across Suggestion Timing, we used multivariate adaptive regression splines (MARS) to model the relationships between suggestion timing and response time.

A.4.2 Response Rate. Response rate was defined as the likelihood that participants accepted a correct suggestion. Significance testing was not applied because the "rate" variable was only meaningful if we considered multiple data points.

A.4.3 Delayed Time. Delayed time was the time delay that was incurred due to incorrect suggestions. Similar to response time, an arcsinh transformation as suggested by the `bestNormalize` package, was applied and a linear mixed model was used to identify significant interaction effects between Task Type and Suggestion Method with regard to Suggestion Timing. The results indicated a significant effect of Task Type × Suggestion Timing ($F = 5.30, p = .021$), but not Suggestion Method × Suggestion Timing ($F = 1.24, p = 0.267$) nor Suggestion Method × Task Type × Suggestion Timing ($F = 0.01, p = .928$).

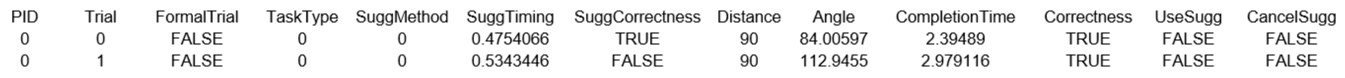
B STUDY 2 - SIMULATION

B.1 Target Prediction Model Mock-up

B.1.1 Target Prediction Model Observations. A selection prediction model based on the available data [20] was replicated and we observed how the predicted probability of the most likely object changed as the task progressed. Further, we drew inspiration from existing research on gaze-based target prediction [15, 36]. From these explorations, we made the following observations:

- The *global centerline* of model confidence over time (i.e., the average trend across all trials) seems to be a sigmoid-like curve [14, 15, 20, 74]. Intuitively, model confidence accelerates from a low point and becomes steady as it approaches an asymptote.
- By replicating [20] and observing results in [36], we found that while the *local centerline* of the model confidence value (i.e., the general trend of each trial) seems to roughly follow a sigmoid-like curve, it can deviate from the global centerline. While the local centerline can still be approximated by a sigmoid curve, the speed of increase can differ on each trial.
- The final confidence curve of each trial, rather than the general trend, contains seemingly randomly-distributed deviations (i.e., spikes and dips) from the local centerline. The evidence was found by replicating [20] and observing results in [36].

B.1.2 Mock-up Prediction Model Generation. Based on these observations, the following trial generation process was formulated for



PID	Trial	FormalTrial	TaskType	SuggMethod	SuggTiming	SuggCorrectness	Distance	Angle	CompletionTime	Correctness	UseSugg	CancelSugg
0	0	FALSE	0	0	0.4754066	TRUE	90	84.00597	2.39489	TRUE	FALSE	FALSE
0	1	FALSE	0	0	0.5343446	FALSE	90	112.9455	2.979116	TRUE	FALSE	FALSE

Figure 11: Example data trials from session 2.

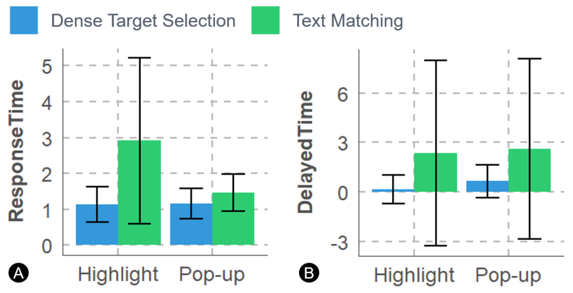


Figure 12: Average response times and delayed times for the suggestion methods (highlighting and pop-up notification) and task types (dense target selection and text matching). The error bars represent $mean \pm std$.

our mock-up prediction model. Our goal was to produce reasonable model confidence curves that mimic an actual prediction model.

- When starting to generate a data trial, the model first samples a trial length t_{max} based on the log-normal distribution regarding user task completion time found in Study 1 (Figure 6A). This sampling approach allows the final dataset to approximate the distribution of user task completion time.
- The model then generates a global centerline based on a sigmoid function $y_1 = sigmoid(x, k, x_0, u, l)$ where k is the logistic growth rate, x_0 is the sigmoid's midpoint, u is the upper bound, and l is the lower bound (Equation 7). This simulates the observation that the global centerline follows a sigmoid curve in an actual prediction model (Figure 6B).

$$y_1 = \frac{u - l}{1 + e^{-k(x - x_0)}} + l \tag{7}$$

- To simulate the variances in a local centerline, the model generates a Bell curve $y_2 = bell(x, \mu, \sigma)$ (Equation 8) to define the area of deviation (see Figure 6C). The distance between the local centerline y_3 and the global centerline is probabilistically sampled from a Gaussian distribution following Equation 9, where μ_r and σ_r are the predefined mean and standard deviation of a Gaussian distribution. By generating random numbers from a Gaussian distribution (with random.gauss), it is more likely that a local centerline is close to the global centerline than further away.

$$y_2 = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{8}$$

$$y_3 = y_1 + y_2 \cdot \text{random.gauss}(\mu_r, \sigma_r) \tag{9}$$

- The final step of the mock-up model is to generate spikes and dips based on the local centerline. To achieve this, the model uses a pre-determined probability j_p to represent the likelihood of jumping to another randomly generated local centerline (new y_3) at a particular timestamp t. The model goes through all timestamps in the trial and modifies the curve depending on it a jump will occur. The resulting curve preserves the property of previous steps: by averaging all generated trials, the centerline still follows a sigmoid function and the local centerline deviates within a predefined region. The model further corrects all probabilities larger than 1 to 1 and smaller than 0 to 0. A sample of a generated trial can be found in Figure 6D.

B.1.3 Dataset Generation. We pre-defined the parameters for the trial generation in later analyses. For the global centerline-related parameters, we set logistic growth rate $k = 2$, sigmoid's midpoint $x_0 = t_{max}/2$, upper bound $u = 1$, lower bound $l = 0$. This simulated a model that knew little information when users started a trial and increased its confidence over time until it reached an almost perfect understanding when users finished the trial, similar to the prediction models in [20] and [36]. Regarding the local centerline-related parameters, we set the bell curve mean $\mu = t_{max}/1.9$ and standard deviation $\sigma = 1$. We also set the Gaussian distribution mean $\mu_r = 0$ and standard deviation $\sigma_r = 1$. The random jump rate j_p was fixed at 0.05. The final results yielded visually similar curves as in the literature [20, 36]. The frame rate was determined to be 50 (0.02 seconds per frame).

B.1.4 RL Reward Settings. Three reward settings were used to train the RL agents. The first reward setting was r_1, where $r_1^t = \texttt{Gain}(t)$ if a suggestion was displayed at t, otherwise $r_1^t = 0$. However, the sparsity in r_1 (i.e., the agent only receives a single reward per trial) prevented many of the agents from learning to display a suggestion at all.

The second reward setting, r_2, sought to solve the reward sparsity issue. Specifically, reward shaping was performed when the suggestion wasn't displayed: $r_2^t = \texttt{Gain}(t)$ if a suggestion was displayed at t, otherwise $r_2^t = -k \cdot p_m$. We used k to penalize the action of not displaying any suggestion. Furthermore, an agent received more of a penalty if it did not display a suggestion when the model confidence value was high (p_m). The penalty factor k was treated as a hyper-parameter during training. While r_2 worked well and enabled the agents to learn to display suggestions, a static value of k might have been limiting. In particular, the penalty of not displaying a suggestion should have changed as training progresses for true reward (i.e., gain function) maximization. In other words, the agent reliance on k should be reduced over the training process. Thus, k was decreased as the training progressed.

The third reward setting also leveraged the benefit of dense rewards, but removed the agents' reliance on the penalty factor k, which may have negative impacts on true reward maximization. In this setting, $r_3^t = \texttt{Gain}(t) - r_3^{t-1}$ (where $r_3^0 = 0$) at a timestamp t. This setting essentially rewarded the agent based on how good it performed on a particularly timestamp t, by computing the contribution of agent's action at t towards the gain. This reward setting thus allowed agents to learn directly from gain functions with dense feedback.

Table 4: Validation and testing results when using Optimal Thresholding and Heuristic Thresholding on the time saved for users and on suggestion usage percentages.

	Task Type	Strategy (Threshold)	Time Saved/Usage% (Std.)	% Improved	Time Saved/Usage%	% Improved
			Validation		*Test*	
Time saved	Dense Target Selection	Optimal Thresholding (0.47)	0.4073s (0.3169s)	44.07%	0.4073s (0.3202s)	39.39%
	Dense Target Selection	Heuristic Thresholding (0.85)	0.2827s (0.3597s)	-	0.2922s (0.3645s)	-
	Text Matching	Optimal Thresholding (0.98)	1.5822s (1.7991s)	268.38%	1.6211s (1.7946s)	260.89%
	Text Matching	Heuristic Thresholding (0.50)	0.4295s (1.1225s)	-	0.4492s (1.1440s)	-
Usage %	Dense Target Selection	Optimal Thresholding (0.81)	65.85% (17.70%)	0.64%	65.69% (18.30%)	0.36%
	Dense Target Selection	Heuristic Thresholding (0.85)	65.43% (20.24%)	-	65.45% (20.42%)	-
	Text Matching	Optimal Thresholding (0.96)	87.33% (18.44%)	50.72%	87.17% (18.53%)	51.52%
	Text Matching	Heuristic Thresholding (0.50)	57.94% (15.85%)	-	57.53% (15.63%)	-

Table 5: Validation and testing results of RL regarding time saved for users and suggestion usage percentages.

Task Type	Strategy	Time Saved (Std.)	% Improved	Time Saved	% Improved	Usage% (Std.)	% Improved	Usage%	% Improved
		Validation		*Test*		*Validation*		*Test*	
Pointing	PPO-MLP	0.4078s (0.3253s)	44.25%	0.4087s (0.3285s)	39.87%	-	-	-	-
Pointing	ACER-LSTM	0.4079s (0.3354s)	44.29%	0.4084s (0.3362s)	39.77%	-	-	-	-
Text Matching	PPO-MLP	1.5673s (1.7878s)	265.91%	1.6050s (1.7877s)	257.30%	87.33% (18.18%)	50.72%	87.31% (18.05%)	51.76%
Text Matching	ACER-LSTM	1.5275s (1.7418s)	240.05%	1.5671s (1.7328s)	248.86%	-	-	-	-

B.1.5 RL Training Methodology. OpenAI Gym [16] with Stable Baselines [34] (for recurrent policies) and Stable Baselines3 [56] (for MLP policies) were used to build and train the RL agents. A preliminary analysis was first run on the toy dataset to determine the appropriate model-free RL training algorithms (PPO2, DQN, A2C, and ACER), reward settings (r_1, r_2, and r_3), policy architectures (MLP and LSTM), policy network size, and training epochs for both task scenarios using the default hyper-parameter settings from the Stable Baselines. This experimentation demonstrated that the PPO2 training with MLP policies was a lightweight and effective solution. ACER with LSTM was the other powerful solution that worked well, but may take longer to train. r_3 was also found to be more suitable for the dense target selection task, while r_2 was better for the text matching task. The training with $4e6$ steps was sufficient for MLP policies and $2e6$ steps was adequate for LSTM policies, based on the convergence of gain in the validation dataset.

After the preliminary exploration, full-range hyper-parameter searches were performed with Optuna [5] using the training dataset for memory size m, penalty k, network size, activation function, learning rate, batch size, discount factor γ, and other algorithm-related parameters following the guidance of RL Baselines Zoo [55]. The model was then fine-tuned by focusing on several key parameters related to training. The training was stopped when the gain in the validation dataset converged. After training all the agents, their performance on the validation and testing dataset were benchmarked.

B.2 Validation and testing results

Detailed validation and testing results of Optimal Thresholding, Heuristic Thresholding, and RL can be found in Table 4 and Table 5.

B.3 Simulation 4: Revisiting a Prior Study

To determine the optimal timing of highlighting suggestions if we were to use an existing model for intelligent suggestion, we ran another simulation using an open-sourced dataset from a prior work [20]. The dataset contained 809 trials with four prediction features over time (i.e., position x, y, z, and rotation yaw every 10 milliseconds) and a final selected target. The original work was replicated with respect to data augmentation, LSTM structure, and training protocol, resulting in a model with 95.06% testing accuracy. For COBO, the features were refit to the trained model to obtain model confidence values over time for the 807 trials.

While it could be challenging to replicate the original study and acquire empirical data on participant response behavior towards intelligent suggestions, the following assumptions were made for the cost and benefit functions: (1) It would take participants 0.5 seconds (i.e., 0.25 seconds reaction time and 0.25 seconds trigger pressing time) to respond to a correct suggestion; (2) An incorrect suggestion would cause 0.25 seconds (i.e., reaction time) of delay; (3) participants would act rationally [62] and would not use a suggestion if the estimated response time (current time + 0.5 seconds) was larger than task completion time of that trial without any suggestion.

Under these assumptions, the optimized threshold for the two objectives were calculated using the COBO framework. The results show that the optimized threshold for completion time ($thres = 0.90$) was able to save 0.0801 seconds ($std.$ = 0.1540 seconds) and the optimized threshold for the usage percentage ($thres$ = 0.82) led to 52.27% ($std. = 42.20\%$) of clicks. Nine Pareto optimal values were also found($thres = 0.82 - 0.90$). The performance improvement in terms of time savings was small for this selection task, although a higher suggestion usage percentage could lead to better user experiences. The original authors' estimate based on the prediction accuracy alone ($thres = 0.85$) was close to our simulation results.