A Meta-Bayesian Approach for Rapid Online Parametric Optimization for Wrist-based Interactions

Yi-Chi Liao
yi-chi.liao@aalto.fi
Aalto University
Helsinki, Finland
Saarland University
Saarbrücken, Germany

Ruta Desai
rutadesai@meta.com
Fundamental AI Research (FAIR), Meta Inc
Redmond, Washington, USA

Alec M. Pierce
alec.m.pierce@gmail.com
Reality Labs Research, Meta Inc
Redmond, Washington, USA

Krista E. Taylor
kristatmail@gmail.com
Reality Labs Research, Meta Inc
Redmond, Washington, USA

Hrvoje Benko
benko@meta.com
Reality Labs Research, Meta Inc
Redmond, Washington, USA

Tanya R. Jonker
tanya.jonker@meta.com
Reality Labs Research, Meta Inc
Redmond, Washington, USA

Aakar Gupta
aakar.hci@gmail.com
Reality Labs Research, Meta Inc
Redmond, Washington, USA
Fujitsu Research America
Pittsburgh, Pennsylvania, USA

ABSTRACT

Wrist-based input often requires tuning parameter settings in correspondence to between-user and between-session differences, such as variations in hand anatomy, wearing position, posture, etc. Traditionally, users either work with predefined parameter values not optimized for individuals or undergo time-consuming calibration processes. We propose an online Bayesian Optimization (BO)-based method for rapidly determining the user-specific optimal settings of wrist-based pointing. Specifically, we develop a *meta-Bayesian optimization (meta-BO)* method, differing from traditional human-in-the-loop BO: By incorporating *meta*-learning of prior optimization data from a user population with *BO*, meta-BO enables rapid calibration of parameters for new users with a handful of trials. We evaluate our method with two representative and distinct wrist-based interactions: absolute and relative pointing. On a weighted-sum metric that consists of completion time, aiming error, and trajectory quality, meta-BO improves absolute pointing performance by 22.92% and 21.35% compared to BO and manual calibration, and improves relative pointing performance by 25.43% and 13.60%.

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642071

CCS CONCEPTS

• **Human-centered computing** → **Pointing devices**; **Interaction techniques**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Bayesian optimization, human-in-the-loop optimization, meta-Bayesian optimization, meta-learning, wrist-based interaction, calibration, target selection, adaptive interface.

ACM Reference Format:
Yi-Chi Liao, Ruta Desai, Alec M. Pierce, Krista E. Taylor, Hrvoje Benko, Tanya R. Jonker, and Aakar Gupta. 2024. A Meta-Bayesian Approach for Rapid Online Parametric Optimization for Wrist-based Interactions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 38 pages. https://doi.org/10.1145/3613904.3642071

1 INTRODUCTION

In addition to its ubiquity in the HCI literature [2, 10, 29, 43, 63, 64], wrist-based input has been posited in the industry as one of the key candidates for driving future freehand interactions in AR[1]. Akin to conventional interaction devices such as a mouse, which are characterized by parameters like transfer function [52] and input filter variables [12], wrist-based devices also possess various parameters for setting transfer functions or device calibration or other input functionalities [11]. The parameter settings for the interactions with these input devices significantly influence the user experience and performance [13, 101]. However, the one-design-fits-all parameter setting strategy of traditional input devices (e.g., mouse,

[1]https://about.fb.com/news/2021/03/inside-facebook-reality-labs-wrist-based-interaction-for-the-next-computing-platform/

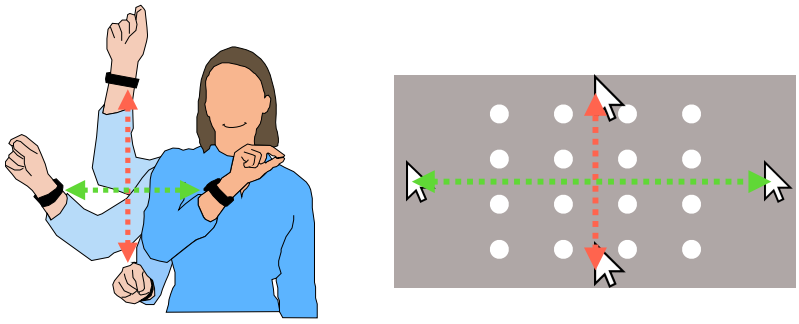

Figure 1: Our absolute pointing interaction using a wrist-worn device. The sensed yaw (green) and pitch (red) values map to the cursor's x and y coordinates.

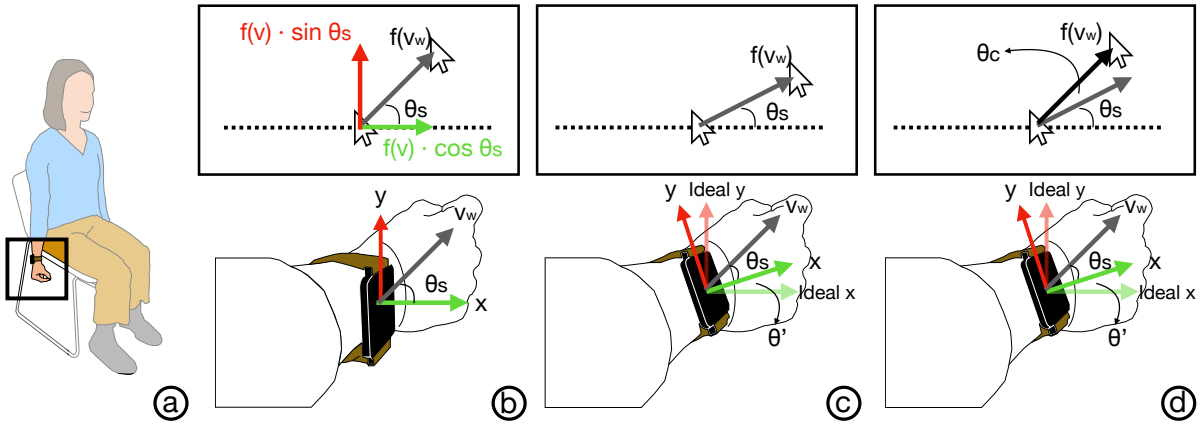

Figure 2: Our relative pointing interaction. (a) Our study participants performed relative pointing in an arms-down position. (b) An ideal situation where the sensed wrist angle θ_s is aligned with the user's intention. (c) A more realistic scenario where the device is worn with a subtle rotational difference θ', hence the sensed direction θ_s is less than the user's intended direction. (d) We use an angular correction parameter θ_c to compensate for such rotational difference. As described in Equation 8, θ_c is added upon θ_s.

keyboard, touch-button, etc.) does not suffice for wrist-based devices for several reasons. Unlike other interaction techniques, a good parameter setting for wrist-based interactions varies significantly across different users owing to their physical, behavioral, and preference-based differences [29]. For example, some users prefer to perform wrist motions with a wider range, while others prefer narrower wrist motions. Further complicating matters, the optimal parameter setting for a user may change based on how the device is worn, hand positions, and environmental factors. A user may wear the device in slightly different positions each time, necessitating unique parameter settings for each use. To identify the best user-specific and session-specific parameter values, users typically undergo a *manual calibration* process to determine a reasonable setting [28, 29]. However, calibration requires extra dedicated time from users, delaying the intended interaction [1, 29, 32, 44, 96], and is often designed manually by developers which does not guarantee the optimal outcome.

Human-in-the-loop (HitL) optimization [31, 85] has the potential to automatically identify optimal parametric settings while users engage in the intended interaction [15, 45, 104]. Among various optimization methods, *Bayesian optimization (BO)* has gained popularity for HitL applications due to its versatility and effectiveness [8, 17, 31, 49, 103]. Although BO has proven to be more sample-efficient than many other algorithms [8], it still requires many iterations and a long duration to converge. For instance, Chan et al. [15] spent an hour optimizing a pointing interaction using BO.

How can we enhance the efficiency of BO for its application in online and rapid human-in-the-loop applications? Our solution augments BO with *prior experience*, enabling it to proactively explore parameter areas with the most potential for promising user performance. We adopt our solution from meta-Bayesian optimization (meta-BO) [3, 23, 54, 91, 93, 100], an emerging paradigm of

BO-based methods for rapid optimization by utilizing datasets gathered from similar optimization tasks[2]. By leveraging the experience of optimizing the parameter settings for a group of users in advance, meta-BO can efficiently optimize for new users for the same interaction. While meta-BO has gained growing attention in machine learning research, its potential utility in HCI remains largely unexplored. We extend a specific meta-BO method called Transfer Acquisition Function (TAF) [100], which builds independent *population models* using the past optimization data and then combines these population models with incoming data for rapid optimization of interaction parameters in online settings. Our approach, which we term *Transfer Acquisition Function$^+$ (TAF$^+$)* extends TAF for parametric optimization in interaction applications by enabling designers to control weighting both in balancing an application's multiple objectives (e.g. speed and accuracy in pointing) and between prior population data and new incoming data in real-time deployments. TAF$^+$ thus enables design flexibility for achieving diverse objectives and efficient adaptation.

We demonstrate the efficacy of our TAF$^+$-powered workflow for rapid parametric optimization of two common and distinct wrist-based pointing interactions: *absolute pointing* (Figure 1) and *relative pointing* (Figure 2). Absolute pointing involves moving the cursor using forearm motion sensed by a wrist inertial measurement unit (IMU) similar to prior work [34, 65]. Relative pointing involves moving the cursor using relative wrist rotation [28, 43, 77]. Pointing is a particularly relevant challenge for AR interfaces, and absolute and relative pointing represent the two fundamental types of pointing. Combined with the significance of wrist-based input for future AR interfaces, wrist-based absolute and relative pointing interactions are a timely and relevant problem to solve.

We conduct a study that evaluates our meta-BO method TAF$^+$ along with two baseline procedures we mentioned above: manual calibration and standard Bayesian optimization. The results show that meta-BO led to significantly better performance on a weighted-sum metric that consisted of completion time, aiming error, and trajectory quality. Specifically, meta-BO improves absolute pointing performance by 22.92% and 21.35% compared to BO and manual calibration respectively, and improves relative pointing performance by 25.43% and 13.60% than BO and manual respectively. In summary, we make the following key contributions:

- *Introducing a novel meta-BO method called Transfer Acquisition Function$^+$ (TAF$^+$) as an online, sample-efficient parametric optimization approach in HCI*: TAF$^+$ extends TAF by enabling designer control of (a) weighting multiple task objectives, and (b) tuning the importance between prior population and current real-time user's data.
- *Demonstrating the efficacy of TAF$^+$ to identify user-specific, optimal parametric settings for two distinct forms of wrist-based pointing*: TAF$^+$ outperformed established baselines - manual calibration and standard BO.

[2]Note that in the context of HitL optimization, "one optimization task" refers to optimizing for "a specific user".

2 RELATED WORK

2.1 Wrist-based interactions

Inertial measurement units (IMUs) are used commonly to detect hand motion. Dipietro et al. [20], Perng et al. [72] proposed various glove-mounted IMU systems, and later research proposed wrist-worn form factors [2, 10, 43, 63, 64]. Previous works have deployed IMUs for detecting human activity [2, 86], projecting raycasting with the detected motion [33, 34, 65, 71], and gesture recognition [51, 59]. Our *absolute pointing* approach is similar to Nancel et al. [65] where forearm movements sensed by an IMU control a cursor's position in a 2D interface. Our *relative pointing* approach uses wrist rotations to control a cursor's relative motion. Wrist rotation tracking via the wristband has typically used outside-in or inside-out sensing. Outside-in uses sensors such as EMG [40, 79], electrical impedance tomography [109], and capacitive sensing [75] for inferring wrist angles or gestures. Inside-out sensing uses wrist-worn cameras [29, 43, 106]. For our relative pointing, we use a similar device and method as RotoWrist [77], which consists of a wrist-worn IR sensor array that tracks the wrist's relative angles.

2.2 Calibration

In HCI, calibration is a procedure for setting up system parameters so the device interaction can work properly, such as tuning a sensor's internal values [25, 74, 105] or setting parameters based on user-dependent features [16, 53]. Calibration has been widely applied, such as for touchscreen interactions [53], gaze input [73], and wearable devices [58, 102]. However, efficient calibration for the transfer function of input devices remains a significant challenge in HCI due to the vast parameter space [52].

Given our focus, we review procedures pertaining to wristband device calibration and pointing transfer function calibration. Wrist-based pointing often needs to identify a *function that maps sensor values to cursor position*. WristWhirl [29] requires users to move their wrists to the maximum along two axes, and then defines a mapping accordingly. Similarly, WristText [28] requires several wrist rotations to capture the maximum sensor values. Our *manual calibration* baseline for *absolute pointing* uses a similar approach where the users identify their preferred operation ranges via forearm motion. While *absolute pointing* involves a one-to-one mapping between input and output (cursor) displacement, *relative pointing* (e.g. mouse pointing or video game sensitivity) involves a CD-gain-based transfer function that varies cursor motion speed based on input motion speed [11, 52]. This function is typically pre-determined using trial-and-error [11] or heuristic iteration [52, 108] and is uniform across all users, with an option for the user to finetune it themselves (as in Windows, Mac, and Linux devices). Our *manual calibration* baseline for *relative pointing* uses a similar approach where we instruct the users to finetune the velocity transfer function starting with a fixed predetermined setting.

2.3 Human-in-the-Loop optimization

Since such calibration procedures do not necessarily result in optimal settings, Human-in-the-Loop (HitL) optimization approaches have been proposed. HitL optimization is a general framework in

which users serve as the evaluation function, and a mathematical parametric optimization procedure aims to efficiently identify the optimal parameter setting [17, 31, 49, 103]. Among many optimization methods, Bayesian optimization (BO) is a computational method that has been developed over decades [62, 84]. Prior work has used BO as an HitL method [39, 47, 76] for minimizing temporal error [57], increasing animation realism [9], game engagement [41], haptic distinguishability [56], hearing aids [68, 69], and optimizing pointing transfer functions [15]. However, BO is mainly seen as an offline design tool [48, 56, 104] since it requires a large number of iterations resulting in long-duration user sessions. For a 3D target-selection transfer function optimization Chan et al. [15], BO required participants to spend 60-90 mins. Consequently, no prior work has employed BO for real-time target selection interactions with end-users. Another recent HitL approach, AutoGain [52], proposed updating the transfer function based on submovement errors. However, AutoGain requires a dedicated session of 30 minutes to converge and is constrained to the single objective of minimizing aiming error. Our meta-BO method is aimed at overcoming these issues and performing rapid, online multi-objective optimization.

2.4 Meta-learning for BO

Meta-learning is a general concept that aims to improve the learning speed of a system by leveraging the prior experience of similar tasks [46, 82, 92, 94]. Successful meta-learning implementations have been proposed for recognition and reinforcement-learning tasks using deep neural nets [6, 21, 26, 67, 78, 81].

In the context of BO, meta-Bayesian optimization (meta-BO) is a machine-learning paradigm consisting of different implementations that use prior optimization data to improve the speed of ongoing optimization [3, 18, 54, 95]. Owing to its recency, meta-BO has not been employed yet to solve HCI problems. Given its promise of speeding up standard BO, we employ the use of meta-BO for our problem. There are multiple ways to apply meta-BO and our goal was to select one that would fit for HitL tasks like ours. The first approach is to fit all the prior data into a unified Gaussian Process (GP) model [5, 7, 91, 107]. However, the model-fitting increases cubically ($O(n^3)$) with the number of observations making the computation time for suggesting the next design impractically long. Incorporating the sparse GP potentially allows for better scalability by only retaining a smaller representative dataset [88, 97]. However, new challenges and uncertainties arise from constructing such complex techniques; for instance, determining the appropriate number of datapoints for the sparse GP, tuning hyperparameters effectively, managing increased model complexity, and balancing computational efficiency with accurate uncertainty estimates. The second approach is to replace elements in BO with neural networks trained on prior data [36, 89, 90, 93, 99]. However, it potentially requires a relatively larger amount of data to pre-train the networks. Moreover, it generally does not offer explanability, which is crucial for HitL applications. The final approach, the one that we adopt, is the weighting-based solution [55, 80, 100] which stores separate GP models, each model being derived from the data of a previous task (in our case, a participant session). The next design suggestion is decided based on a weighted aggregation of all the previously gathered GPs' information. This approach has low computational complexity, can work with small amounts of prior data, and offers higher explainability to the users with the opportunity of observing the result generated by each model. Among several implementations along this line of research [24, 37, 55, 80], our method extends Wistuba et al. [100]'s TAF approach. They demonstrate TAF with a single-objective and a naive test function. Our TAF+ approach extends this to our multi-objective HitL scenario which presents new challenges.

3 PRELIMINARIES: EXISTING METHODS

To appropriately explain TAF$^+$, we first introduce the key concepts for BO and meta-BO in this section. Given that TAF$^+$ is built upon an existing meta-BO method called Transfer Acquisition Function (TAF), we also provide an overview of TAF and its limitations.

3.1 BO using Expected Improvement as the acquisition function

BO identifies the optimal parameter setting that maximizes or minimizes an objective function f (e.g., completion time) over iterations. In each iteration, BO selects a parameter setting (denoted as x) to be evaluated ($f(x)$), resulting in an objective function value $y = f(x)$. BO has two key elements: the *acquisition function* determines which x should be evaluated in each iteration, and the *surrogate model* of the true objective function f.

Since each evaluation of f is expensive e.g., through user interaction, BO relies on *acquisition functions*, which is cheaper to evaluate [38], to determine the next setting to be evaluated. In each iteration, BO samples many parameter settings and calculates their *acquisition values* (its "worth value") with the acquisition function. The x with the highest acquisition value is picked for the actual evaluation. To generate the acquisition value of a given x, the acquisition function relies on BO's another element — the surrogate model. This surrogate model is typically a Gaussian Process regression (GP) [70, 83], which generates the predicted objective value ($\hat{y}$) and its associated uncertainty (i.e., variance)[3] for a given x. After the actual evaluation of f, each BO iteration results in an observation of (x, y) pair; all the observations are then fit into the surrogate model (i.e., GP). As BO accumulates more data through iterations, its GP becomes more accurate to the true function f, enabling the acquisition function to make better predictions. For more details on BO, please see [27].

Among common acquisition functions [27, 98], we select *Expected Improvement (EI)* as the base acquisition function since it is used in the TAF paper, upon which we develop our TAF$^+$ algorithm. An intuitive way to understand *Expected Improvement* ($EI(x)$) is that it calculates the amount of potential improvement in the objective function from the current best observation. A formal definition is:

$$EI(x) \equiv \mathbb{E}_{f(x)}\{\max[\hat{f}(x) - f^+, 0] \mid \mathbb{H}\}, \tag{1}$$

where $EI(x)$ is the acquisition value at x for the current iteration t, $\hat{y} = \hat{f}(x)$ is the predicted objective value at x based on the GP model, f^+ is the best-observed performance thus far over the

[3]Note: when dealing with human-in-the-loop tasks, it is recommended to incorporate the GP with the inferred noise levels such as the Botorch's single-task GP implementation: https://botorch.org/docs/models.

whole optimization history $\mathbb{H}$, consisting of all previous datapoints $\{(x_1, y_1) \dots (x_{t-1}, y_{t-1}))\}$.

3.2 Meta-BO

Meta-learning is a paradigm of machine learning, focused on achieving fast adaptation in a given task by *leveraging prior data of similar tasks* [35]. In the context of HitL BO, each task is to identify the optimal parameter setting of a user using BO. Thus, the goal of meta-learning for BO (i.e., meta-BO) is to leverage the optimization data of previous users to enhance the efficiency of BO for the new user(s). There are generally two phases in meta-BO (Figure 3): The first phase is *population modeling* which involves gathering data from a set of users. We run HitL BO on each user, resulting in one GP model (i.e., surrogate model) per user. We define these models as *population models*. In the second phase, *adaptation*, meta-BO is deployed on the new users. In particular, a new GP is constructed by fitting the observations of the new user, while leveraging the population models when possible. We call this new GP model *adaptation model* since it aims to "adapt" the previous GP models to the new user.

3.3 Transfer Acquisition Function (TAF): a meta-BO method

BO needs to search randomly in the initial iterations since its surrogate GP model does not have information to guide a meaningful search. *Transfer Acquisition Function (TAF)* is a specific meta-BO method that addresses this limitation by utilizing previous population models as informative prior for guiding the search. Specifically, TAF is an acquisition function that considers both the adaptation model built upon the observations of the current user and the previously gathered population models. Thus, even when the adaptation model has no (or limited) information, population models can still guide the optimizer in selecting a setting (x) that is likely to lead to good performances (y). Similar to the purpose of the regular acquisition functions in the BO process, the x with the highest *TAF value* will then be evaluated by the new user in each iteration. The gathered observation will fit into only the adaptation model, further improving its predictions in the later iterations.

TAF aggregates the Expected Improvement (*EI*) value from the current adaptation model as well as from all the population models for a given parameter setting x. Here we define the *EI* calculated from the population models as *Population Expected Improvement (PEI)* to differentiate from the *EI* calculated from the current adaptation model. *PEI* is calculated similarly to *EI* (see Equation 1, and refer to section 7 of the original paper [100] for more details). This leads to one *EI* value and M *PEI* values for a given x, where M is the number of population models. TAF then computes a weighted combination of these values to obtain the acquisition function value at x:

$$TAF(x) = \frac{\sum_{j=1}^{M} w_j PEI_j(x) + w_{M+1} EI(x)}{\sum_{j=1}^{M+1} w_j}, \tag{2}$$

where $EI(x)$ is the Expected Improvement from the adaptation model, while $PEI_j(x)$ is the Population Expected Improvement based on the j-th population model. $j \in 1, ..., M$ denotes the index of the population models. Finally, w_j are weights on each *PEI*, and w_{M+1} is the weight assigned on the *EI*. We define this summary across different models as **"between-model combination"** (Figure 3).

3.3.1 Model weights. TAF computes a weighted combination of the *EI* and *PEI* values using weights w_j. A higher weight means that this model's *EI* (or *PEI*) value is more valuable or reliable. Here, we define such weights on models as "model weights". Following Wistuba et al. [100], we use a variance-based method for determining these weights. An intuitive explanation is that the weight of a model is based on the *confidence* of its prediction at x. Low variance from a particular model (see the red area of each model in Figure 3) indicates higher confidence, so its resulting weight is higher in TAF computation (Equation 2). On the contrary, when the variance is large, the model has less confidence in its prediction, so the corresponding weight should be lower[4].

The population models generally do not have a large variance since they are already fitted with data from the previous optimization processes. However, at the beginning of an adaptation, the adaptation model has none or very few datapoints, so the variance is overall high, leading to its low model weight. Hence, TAF initially relies more on the population models. Figure 3 shows an example. As the adaptation model is fitted with more observations from the current user, its variances decrease over iterations. Consequently, TAF gradually increases the adaptation model's weight after ample iterations, achieving personalization.

3.3.2 Limitations of TAF. Prior works have evaluated the performance of TAF with various single-objective testing functions. TAF significantly outperformed the standard BO while being computationally lightweight [93, 100]. However, TAF has two major limitations, making it unsuitable for realistic HitL problems. It was designed to handle a single objective. Yet, a realistic interaction usually involves multiple objectives, which are unclear as to how to address with TAF. Secondly, TAF does not allow proactively shifting weights from population models to the adaptation model. Although TAF gradually increases the weights of the adaptation model with more observations, there are scenarios in which we hope TAF relies on the adaptation model in earlier iterations. For example, when the new user exhibits behaviors different from all the population models, shifting the importance to the adaptation model allows for a more efficient user-specific adaptation. However, TAF does not have a mechanism to support that.

4 TRANSFER ACQUISITION FUNCTION$^+$ (TAF$^+$) AND ITS ACCOMPANYING WORKFLOW

We develop our Transfer Acquisition Function$^+$ (TAF$^+$) algorithm by leveraging TAF. Accompanying the algorithm is its workflow (see Figure 5). The first three steps are offline steps to be performed by the developer before deploying the method on the device. The

[4]Please refer to Section 6.1 of the original paper [100] for details, in which this particular weighting method was named TAF-M. The same implementation was replicated in Volpp et al. [93], where it was referred to as TAF-ME.

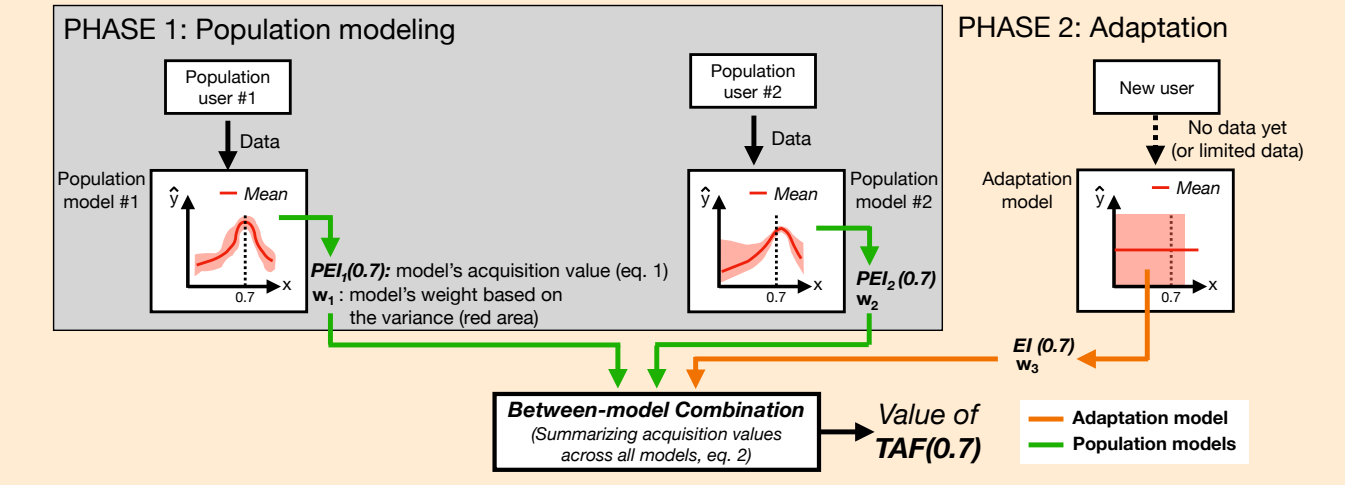

Figure 3: Overview of Transfer Acquisition Function (TAF) with an example of 2 population models: TAF is a weighted sum of several acquisition values generated by the currently constructed model and the models gathered in advance (i.e., EI or PEI_j where j is the index of the models). In PHASE 1, Population models are constructed per user using optimization data. Each model predicts the user performance $\hat{y}$ (red line), the corresponding acquisition value (PEI), and the uncertainty of this prediction (the red area) of a specific x. In PHASE 2, a new Adaptation model is created for the new user. To derive the acquisition value, TAF computes *between-model combination* values across all models (including adaptation model's EI and population models' $PEIs$) based on the *model weights*. Model weights are denoted as w_j, and they are computed based on the variance (width of the red area) of each prediction. The EI (or PEI) with a higher uncertainty have lower weights. The example is computing the TAF at $x = 0.7$, where the adaptation model has very high uncertainty in early iterations, so the TAF value is majorly determined by the population models. As the adaptation model gains more observations, TAF will gradually be dominated by the Adaptation model, leading to the user-specific optimal result.

final step is the online adaptation that the end-users experience while performing the pointing interactions with the device.

Here, we first introduce TAF$^+$ and then provide an overview of each step of the workflow. Lastly, we compare the TAF$^+$ workflow with other baseline methods.

4.1 Transfer Acquisition Function$^+$ (TAF$^+$)

The main idea of TAF$^+$ is to mitigate the limitations of TAF with two crucial extensions: *dynamically handling multiple objectives* and *proactively balancing the weights of the population models and the adaptation model.*

4.1.1 Extension 1: Dynamically handling multiple objectives. In realistic interactions, there is usually more than one objective function, which limits the direct application of TAF. A naive solution is a weighted-sum approach that transforms multiple objectives into a single objective, where a set of weights on each objective must be predefined. The resultant weighted-sum objective can then be used for both *population modeling* and *adaptation* phases, thereby enabling the application of TAF to multi-objective settings. Despite its simplicity, this approach has various limitations for realistic tasks. The weight assignment needs to be arbitrarily done by the designer beforehand, and there is no flexibility to tune the weights later. In practice, there is a high potential that the designer would need to adjust the weights since the predefined weights may not be ideal.

As shown in Figure 4, our TAF$^+$ takes a different approach. Instead of predetermining a fixed set of weights for objective functions, our population modeling is performed in a multi-objective manner. That is, our population models generate multiple acquisition values, each for one objective, instead of only a single value, for a given x. Then, in the adaptation phase, TAF$^+$ *dynamically* combines these acquisition values into one value per model according to the weights assigned to the objective functions. Such weights can be adjusted whenever needed.

We denote the acquisition values on different objectives as EI^i or PEI^i, where i is the index of the objectives. During the adaptation phase, for each model, TAF$^+$ first combines these values of different objectives into a single acquisition value (EI^+ or PEI^+, where the + sign indicates this is an aggregated value) based on the weights of objectives. We refer to this combination as **"within-model combination"** to differentiate from the between-model combination (Figure 3), and we define these weights of the objectives as "OBJECTIVE WEIGHTS" to differentiate them from the weights on models. This feature enables a designer to dynamically adjust the OBJECTIVE WEIGHTS at any time, even after the population modeling. Designers can even dynamically tailor the OBJECTIVE WEIGHTS for different users or contexts. In subsection 4.2, we illustrate a workflow that allows designers to identify the optimal OBJECTIVE WEIGHTS based on the users' subjective ratings (see step 2 in Figure 5). Subsequently, TAF$^+$ combines all the acquisition values across different models into a final value based on the *model*

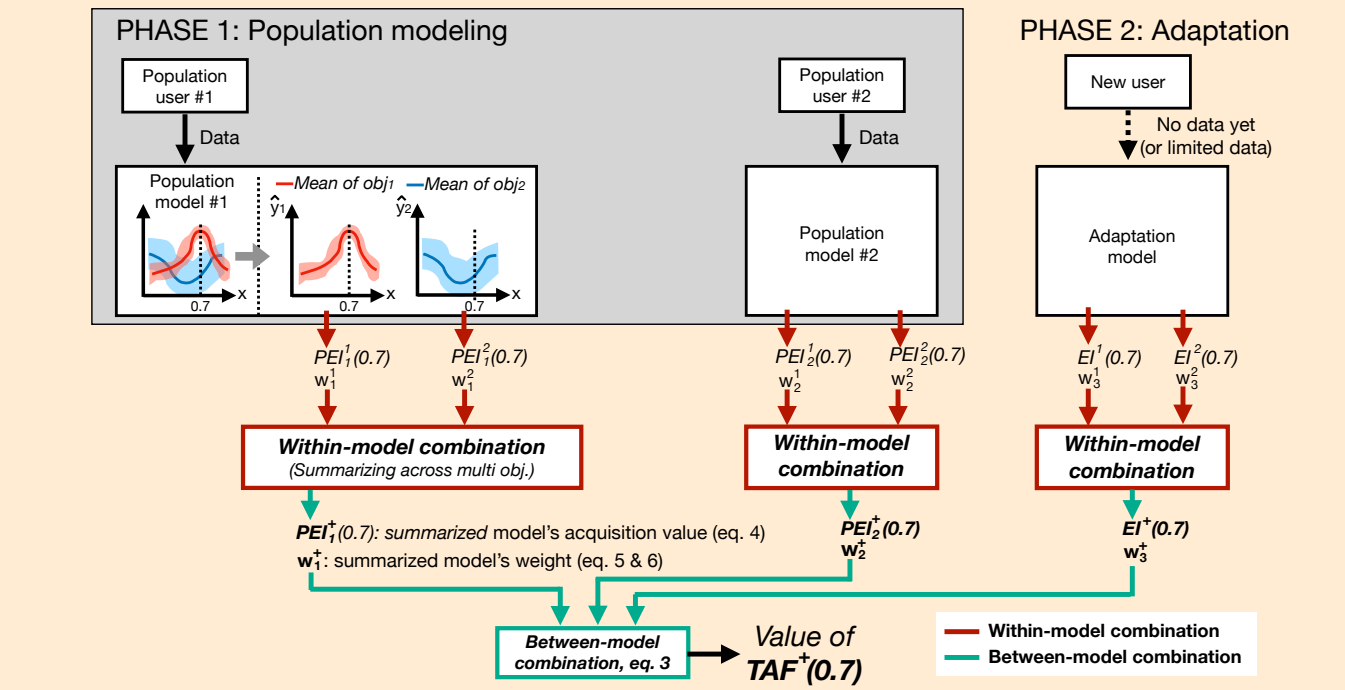

Figure 4: Transfer Acquisition Function$^+$ (TAF$^+$) in an example task of two objectives: Both the Adaptation model and the Population models can generate predicted performances of two objectives (red/blue lines), the corresponding acquisition values (EI^i or PEI^i_j, where i is the index of the objectives and j is the index of the models), and the uncertainty of this prediction (shown as the red/blue are). The two major phases (population modeling and adaptation) are identical to TAF (Figure 3). For handling multiple objectives, TAF$^+$ first performs a *within-model combination*: Within a model j, it summarizes acquisition values (EI^i_j or PEI^i_j) of different objectives into a single weighted-sum value (EI^+_j or PEI^+_j) and summarizes weights of different objectives (w^i_j) into a single weight (w^+_j) based on the OBJECTIVE WEIGHTS. Once every model has a summed acquisition value (EI^+ or PEI^+_j) and a MODEL WEIGHT (w^+_j), TAF$^+$ then performs a *between-model combination*, similarly to TAF, deriving the final TAF+ value.

weights. Such a **between-model combination** is identical to that in TAF. To this point, TAF$^+$ can be formally written as:

$$TAF^+(x) = \frac{\sum_{j=1}^{M} w^+_j PEI^+_j(x) + w^+_{M+1} EI^+(x)}{\sum_{j=1}^{M+1} w^+_j}, \tag{3}$$

where EI^+ and PEI^+ are weighted sums of EI and PEI values respectively from several objectives. $j \in 1, ..., M$ denote the index of the population models. Assuming n different objectives, we can denote EI^+ and PEI^+ as:

$$PEI^+_j(x) = \sum_{i=1}^{n} \alpha^i \cdot PEI^i_j(x), \quad EI^+ = \sum_{i=1}^{n} \alpha^i \cdot EI^i(x), \tag{4}$$

where $i \in \{1, ..., n\}$ denotes the index of objectives, and α^i is the OBJECTIVE WEIGHT of the i-th objective function. Furthermore, in contrast to TAF, where each population or adaptation model associates with a single objective and has one *model weight*, TAF$^+$ considers multiple objectives. Consequently, each model has multiple weights – each is associated with one objective for each EI or PEI. Formally, we denote these weights as w^i where i is the index of the objectives. During the **within-model combination** in TAF$^+$, similar to the process of deriving EI^+ or PEI^+, the OBJECTIVE WEIGHTS α^i are utilized to combine multiple weights w^i into one single weight w^+_j:

$$w^+_j = d(k) \cdot \sum_{i=1}^{n} \alpha^i \cdot w^i_j(x), \quad w^+_{M+1} = \sum_{i=1}^{n} \alpha^i \cdot w^i_{M+1}(x), \tag{5}$$

where α^i is the same OBJECTIVE WEIGHT shared with Equation 4, and $w^i_j(x)$ is the j-th model's weight on the i-th objective function. Each $w^i_j(x)$ value is calculated in a manner similar to TAF, as described in subsubsection 3.3.1. $d(k)$ is a decay factor applied only to the population models, which will be elaborated in subsubsection 4.1.2 and Equation 6.

4.1.2 Extension 2: Proactively balancing the weights of population models and the adaptation model. The second extension is meant to cope with the potentially high user diversity. $d(k)$ in Equation 5 denotes the decay of the weights on the population models. This decay is a scalar value ranging $[0, 1]$, which is directly multiplied by the original weight values $w^i_j(x)$. $d(k) = 1$ means there is no decay,

and $d(k) = 0$ means the adaptation fully relying on the adaptation model. An ideal decay should update over iterations – it may not be effective in the early iterations since the adaptation model has very limited information but should increase over iterations as more data becomes available from the current user. The decay $d(k)$ is thus an iteration-dependent function with two hyperparameters: d_1 the iteration number after which the decay kicks into effect, and the other d_2 determines the rate of the decay. We can formally describe $d(k)$ as:

$$d(k) = \begin{cases} 1, & \text{if } k \leq d_1 \\ 1 - (k - d_1) \cdot d_2, & \text{if } d_1 < k \leq d_1 + \frac{1}{d_2} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where k is the count of the iteration, d_1 is a positive integer (or 0), and $d_2 \in (0, 1]$. There is no decay when the iteration count is less than d_1. After d_1-th iteration, the decay starts with the rate of d_2; i.e., every iteration, the scalar $d(k)$ decreases by d_2. Once the scalar reaches 0, it stays at 0, so w_j^+ stays 0 as well, and hence adaptation starts fully relying on the adaptation model.

With this extension, TAF$^+$ allows actively determining how the population models should decrease their importance. For instance, in a population where all the users exhibit high similarity, the optimal design for the new user is likely to be highly similar to the population models. Hence, the designer can set d_2 as 0, allowing the population models to consistently guide the current optimization. On the other hand, when there is a higher diversity between users, a designer can leverage the population models in the initial steps and let the adaptation quickly develop based on the current data. In such a case, the designer can set d_1 and d_2 to properly decay to achieve faster adaptation. Our workflow has an additional step to derive the optimal d_1 and d_2 (see step 3 in Figure 5). On the contrary, TAF does not have a similar mechanism, potentially not suitable for interactions requiring fast adaptation.

4.1.3 Potential generalizability of TAF$^+$ for HitL optimization: TAF$^+$ shares the foundational principles of BO, a versatile parametric optimization method with minimal assumptions of the task [27, 84] and has proven its generalizability over a wide range of HCI applications [9, 15, 17, 47, 57, 87]. Extended from BO, the only essential assumption of our TAF$^+$ is that while users have individual differences, there exist parameter ranges that generally lead to good user performance. This is a common assumption in HCI and design, where a certain range of parameters or designs are considered effective for the broader user base despite individual disparities. Thus, TAF$^+$ can be used as a general approach for other HitL problems as well. In rare occasions where there are completely no overlapping traits between the population models, designers can use the TAF$^+$ workflow to forecast this outcome (see subsection 4.5) and use other methods instead. We demonstrate the potential generalizability of TAF$^+$ with a series of simulations, as presented in subsection 4.6.

4.2 Overview of the TAF$^+$ workflow

Figure 5 provides the overview of TAF$^+$ workflow, which contains three steps to prepare the population models followed by the deployment in the adaptation phase. The **first step** of TAF$^+$ workflow entails building population models that generate multi-objective predictions. This step eliminates the need to predefine the OBJECTIVE WEIGHTS for multiple objectives in advance and enables the flexibility of setting them later. The **second step** focuses on identifying the optimal OBJECTIVE WEIGHTS α corresponding to the maximum subjective ratings. By optimizing the OBJECTIVE WEIGHTS, the designers can actively steer the optimization to explore the parts of the Pareto-frontier that maximize the user's feedback. In the **third step**, the optimal decay hyperparameter settings (Equation 6) for the gathered population models are identified using grid search in simulations. Different decay hyperparameter settings are tested in simulation to identify which setting leads to the optimal simulated adaptation performance. Finally, we deploy meta-BO, our TAF$^+$, in **adaptation** on the end-users, where the population models, optimal OBJECTIVE WEIGHTS, and the decay hyperparameters are utilized. We detail each step below.

4.3 Step 1: Population modeling

We performed a data collection where users went through HitL optimization guided by multi-objective BO. The data of each user is then used to construct a GP model, which predicts the performance of multiple objective functions when given a x. Note that this step does not incur any additional costs for end-users because the end-users only experience the adaptation phase (see Figure 5).

4.4 Step 2: OBJECTIVE WEIGHT α optimization

TAF$^+$ transforms the optimization problem from multiple objectives into a single objective based on OBJECTIVE WEIGHTS. The selected three objectives involve intrinsic trade-offs, the same as other input devices. To identify the optimal weights, the subjective ratings of each user's Pareto-optimal designs are obtained and used to identify the weight set that results in the designs with the highest subjective rating. This step is a "population-level" weight optimization because its goal is to identify the weight setting that captures the highest ratings across all users.

4.4.1 Objective weight optimization for a single user: Consider a simplified single-user scenario for a better understanding of our method (Table 1). Our procedure involves sampling a list of possible weight sets (e.g, [0.1, 0.1, 0.8], [0.1, 0.2, 0.7], ... for a problem with three objectives). Each weight set is applied to all the Pareto-optimal designs' objective values of this user to identify the design among the Pareto frontier with the highest weighted-sum objective value. We then record the user's rating for this particular design as the score of this weight. By trying out all the weights and comparing their corresponding ratings, we can identify the optimal OBJECTIVE WEIGHTS that leads to the highest user rating.

With a single user, we could conclude the OBJECTIVE WEIGHT assignment by assigning weights in accordance with the highest user rating. However, at the population level, different users may favor the objectives differently. We therefore need to find the OBJECTIVE WEIGHT assignment that is the best across all users in the population. For instance, for the user presented in Table 1, design B has the highest user rating, and its third objective function has the highest value. Intuitively, it suggests that this user favors the third objective; then assigning the OBJECTIVE WEIGHTS as $[0.1, 0.1, 0.8]$ is a reasonable and straightforward solution. However, at the population level, different users potentially favor different objectives, so it

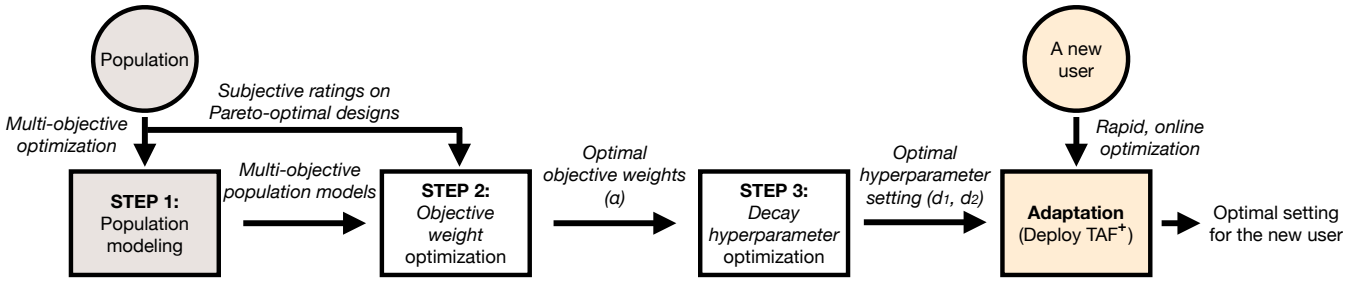

Figure 5: TAF$^+$ workflow: subsection 4.2 provides the details and explanations of each step. Similar to other meta-learning workflows, the first step in ours is population modeling, and finally, TAF$^+$ is deployed on the end-users (adaptation). TAF$^+$ workflow has two additional steps (steps 2 and 3) for deriving the optimal OBJECTIVE WEIGHTS and the decay hyperparameters.

Pareto-optimal designs/settings	Original objective values	Weighted-sum objective value (weights=[0.7, 0.2, 0.1])	Weighted-sum objective value (weights=[0.2, 0.3, 0.5])	Weighted-sum objective value (weights=[0.2, 0.6, 0.2])	User rating
A	[7, 2, 5]	**5.8**	5.5	3.6	75
B	[5, 4, 8]	5.1	**6.2**	5	**100**
C	[3, 9, 2]	4.1	4.3	**6.4**	1

Table 1: The table shows a set of Pareto-optimal designs in a three-objective optimization problem. We demonstrate how to obtain the optimal OBJECTIVE WEIGHTS leveraging the user ratings. The second column shows that designs A, B, and C have different objective value sets. Columns 3 - 5 demonstrate we can calculate the weighted-sum objective values when a set of OBJECTIVE WEIGHTS are given. Different OBJECTIVE WEIGHTS lead to different optimal designs. For instance, under weights = [0.7, 0.2, 0.1], design A is the best design. However, under weights = [0.2, 0.3, 0.5], design B is the optimal design. Since we have gathered the user's ratings on each design, we can compare which OBJECTIVE WEIGHT setting leads to the design that corresponds to a higher user rating. In this example, the second weight ([0.2, 0.3, 0.5]) is the most preferred weight among the three weight settings because it leads to B as the final design, whose user rating is the highest, 100.

is unlikely to intuitively identify a single objective that leads to the highest ratings for all. Therefore, a grid search is a more thorough solution. More details are below.

4.4.2 OBJECTIVE WEIGHT optimization across all users: To obtain the best weight setting across all users, we leverage the process described in subsubsection 4.4.1 for each user in the population. In particular, we obtain the weighted-sum optimal designs among the Pareto-optimal designs of all users by running the aforementioned process for a given weight set (e.g, [0.1, 0.1, 0.8], [0.1, 0.2, 0.7], [0.1, 0.3, 0.6], ...). Then, all the users' subjective ratings corresponding to the resultant optimal designs are combined as a final score for that OBJECTIVE WEIGHT set. Finally, the final scores of all weight sets are compared, and the weight set that leads to the highest net user rating is identified. **Appendix A.1** elaborates on the need for searching for the best OBJECTIVE WEIGHT configuration in this manner through a small example scenario. **Appendix A.2** also provides the detailed algorithm that is explained here.

4.5 Step 3: Decay hyperparameter optimization

TAF$^+$ has a set of hyperparameters that decreases the weights of the population models (Equation 6). The step aims to identify the optimal hyperparameter setting through simulations. In these simulations, we take one population model at a time and treat it as a new user. Meanwhile, we treat the remaining models as population models to conduct TAF$^+$. We then performed a grid search over different sets of $< d_1, d_2 >$ values to identify the $< d_1, d_2 >$ pair that yielded the best performance at the population level. This optimal hyperparameter setting can then be used for subsequent TAF$^+$ runs.

This simulation can be an effective pre-check step before deploying meta-BO. Future practitioners can utilize such simulations to foresee the potential efficacy of meta-BO for their own tasks. In rare cases, users may behave completely differently for a particular interaction. This step informs the practitioners that TAF$^+$ would not outperform BO regardless of the hyperparameter setting, so they can deploy a standard BO instead. Furthermore, practitioners can utilize the simulations to observe the performance of meta-BO with different numbers of population models and learn that larger user groups may be needed in certain cases.

4.6 Evaluating TAF$^+$'s viability via simulations on synthetic test functions

Before applying TAF$^+$ onto the target interactions, we present synthetic simulations that evaluate TAF+ in multi-objective problems with common test functions, such as Sphere[5], Branin[6], and Hartmann 3D[7]. Our simulations use the same objective functions and

[5]See https://www.sfu.ca/~ssurjano/spheref.html.
[6]See http://www.sfu.ca/~ssurjano/branin.html.
[7]See https://www.sfu.ca/~ssurjano/hart3.html.

number of parameters as relative pointing. We evaluated TAF$^+$'s performance under 6 different OBJECTIVE WEIGHT configurations, highlighting its advantage of offering high flexibility for the designers to fine-tune the OBJECTIVE WEIGHT when needed. We further evaluated TAF$^+$'s performance with five levels of user group similarity, showing its effectiveness even when users exhibit high differences. We also evaluated the potential generalizability of TAF$^+$ with four different functions and with five population model sizes. Our simulation analysis shows that TAF+ always converged to global optimality and outperformed BO in various conditions. This provides evidence for its potential in a wide range of interactions. **Appendix B** presents the detailed procedure and results of our simulations.

4.7 Summary

To summarize, TAF$^+$ improves over TAF so as to handle HitL scenarios in two aspects: it can **flexibly handle multiple objectives** and it can **proactively decay the weights of prior models** as the user progresses in the adaptation phase. Further, TAF$^+$ achieves its goal of converging faster than standard BO in the synthetic simulations.

5 INTERACTIONS

We study two distinct and representative wrist-based pointing interactions: *absolute pointing* and *relative pointing*. While these two interactions share the same task and objective functions, they use different hardware (IMU v.s. infrared sensors), different body parts (forearm v.s. wrist angular motion), different device parameters (forearm yaw-pitch v.s. wrist angle), different transfer functions (linear v.s. sigmoid) and different parameter counts (2 v.s. 4). We purposely chose these two cases to demonstrate the effectiveness and the potential generalizability of the meta-BO approach. Below, we first describe the shared details and then the two interactions.

5.1 Task and software interface

5.1.1 Task: Both interactions involved 2D target selection (Figure 6), where participants were asked to move the cursor to the target and select it by performing a double pinch. The pinches were detected using the highly accurate active electrical sensing approach, the same technique used in ElectroRing [42]. Participants were asked to select the targets *"as quickly and accurately as possible"*.

5.1.2 Interface: Our goal is to deploy meta-BO online while the user performs pointing in a real-world interface. We thus design our study interface with varying target sizes and distances in a grid to resemble a real-world interface (see Figure 6 a). The targets are circular, arranged in an 8×4 grid. The diameter of each circle is uniformly sampled from a range of $[20, 35]$ *mm*. The default color of all the circles is light grey, and the target circle is highlighted in blue. Upon selection, the target turns red. The system then randomly samples a different circle as the next target. The cursor does not reset to the origin between selections. We apply a one-euro filter on the cursor position to overcome jitter [12].

5.2 Objective functions

Two interactions share the same objective functions, which we aim to optimize. To prevent bias toward any objective function during multi-objective optimization, these objective functions were further linearly normalized into the range of $[-1, 1]$. We converted the problem into a maximization problem, so 1 is the best performance and -1 is the worst[8]. Similar normalization was also done in prior work [15].

1. Normalized completion time (NCT): Completion time (CT) is the duration from the moment that the cursor leaves the previous target to when the selection is complete, a typical way to assess the input **efficiency**. Our targets varied in size and distance; a standard way to counterbalance the effect of these variances is to average the performance over many selections (e.g., [15]). However, since we aimed to use the minimal number of selections for the optimization process, we normalized the completion time with the Index of Difficulty (ID) [61], and thus $NCT = \frac{CT}{ID}$.

2. Trajectory aiming error (TAE): TAE is a crucial metric for evaluating **pointing accuracy** [15, 52]. We follow AutoGain [52] that used the Persistence1D algorithm [50] to segment a full trajectory into sub-movements and then calculated the aiming error of each ballistic sub-movement. For simplicity, we assumed the implicit aiming point to always be the target position. After excluding the unaimed, interrupted, and non-ballistic movements, we then calculate the aiming error (overshoot and undershoot) of each sub-movement as the distance between the cursor position at the local minimum speed and the closest edge of the target (Figure 6 a). Summing up all the aiming errors (AE_i, where $i \in [1..n]$ is the i-th ballistic sub-movement), $TAE = \sum_{i=1}^{n} AE_i$.

3. Trajectory Travel Distance (TTD): TTD measures the amount of **detour** of a selection trajectory. A selection could be fast and accurate, but if the cursor travel distance from the previous to the current target is longer than the shortest distance, it indicates a potentially skewed transfer function. This metric is expressed as $TTD = D_m/D_i$, where D_m is the measured travel distance and D_i is the ideal distance. In absolute pointing, if the transfer function values misalign with the user's assumed ratio, the user would take extra distance to move the cursor along the intended direction. In relative pointing, a rotational misalignment in the x-y mapping may result in deviations in motion. While TTD may be correlated with the previous objective functions, the correlation is partial. TTD captures additional useful information. Note that there are intrinsic trade-offs in the selected objectives.

5.3 Optimization iteration

From a 4-person pilot test, we decided to have 6 target selections in each optimization iteration. The first two selections were seen as "practice" because the users may still be adapting to the new setting. We took the average value of the three objective functions (see subsection 5.2) of the last four selections as the final objective values of that iteration.

[8]We conducted a 4-participant pilot study to derive the parameters for the linear normalization which ensures three objective functions have similar mean values and ranges.

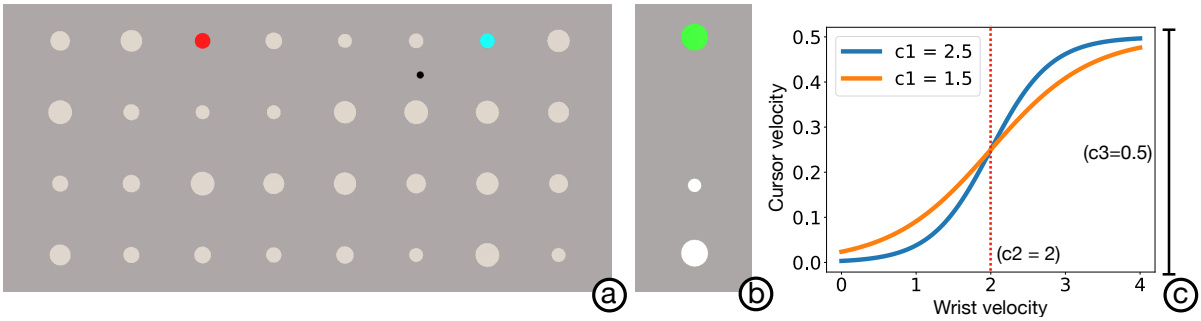

Figure 6: Our study interface: (a) The main study interface. The smaller black dot is the cursor, the red circle is the previous target, and the blue circle is the current target. (b) The interface for calibrating for the relative pointing. The cursor only moves on the y-axis during calibration when the participant performs radial-ulnar deviation motions. (c) Example transfer functions for relative pointing.

Table 2: Design parameterization of absolute pointing. Both scalars have been normalized to the $[0, 1]$ range.

Design Parameter		Range
S_x:	Scalar factor maps yaw to x	$[0, 1]$
S_y:	Scalar factor maps $pitch$ to y	$[0, 1]$

Table 3: Design parameterization of relative pointing.

Design Parameter		Range
θ_c:	angular correction parameter	$[-10°, 50°]$
c_1:	Sigmoid function (overall shape)	$[0.5, 2.5]$
c_2:	Sigmoid function (center x position)	$[0.5, 2.5]$
c_3:	Sigmoid function (overall scale in y axis)	$[0.5, 2]$

5.4 Absolute pointing

Our absolute pointing interaction utilizes an IMU on the wrist to detect *the absolute position of the forearm* (Figure 1). The challenge here is to identify the ideal function that maps the forearm positions to the cursor's 2D positions.

5.4.1 Device and interaction: The interaction linearly maps the IMU $< yaw, pitch >$ to $< x, y >$ coordinates on the interface. Initially, the $< yaw, pitch >$ corresponding to the user's preferred central forearm position in the air is mapped to $< 0, 0 >$. The system then linearly updates the cursor's x & y positions based on yaw & $pitch$ respectively. Ideally, each $< x, y >$ coordinate corresponded to a specific $< yaw, pitch >$ pair.

5.4.2 Design parameters: The transfer functions for determining cursor position are:

$$C_x(t) = C_x(0) + (Yaw(t) - Yaw(0)) \cdot S_x, \\ C_y(t) = C_y(0) + (Pitch(t) - Pitch(0)) \cdot S_y, \tag{7}$$

where $C_x(0)$ and $C_y(0)$ denote the cursor's centered x and y positions in the scene. $C_x(t)$ and $C_y(t)$ stand for the cursor's x and y positions at time t. $Yaw(0)$ and $Pitch(0)$ are the centered Yaw and $Pitch$ values. $Yaw(t)$ and $Pitch(t)$ are the Yaw and $Pitch$ value at timestamp t. The two design parameters that need to be determined are the transfer scalars S_x and S_y (Table 2). For different users, different motion ranges may be optimal (which may also depend on the task), thus requiring per-user parameter optimization.

5.5 Relative pointing

Our relative pointing (Figure 2) interaction is similar to mouse pointing where users move the cursor by clutching [66]. The cursor's direction is determined by the sensed wrist movement's angle. The cursor's moving velocity is determined by a velocity function that takes the wrist's relative velocity as input. The user pinches to initiate cursor motion, rotates the wrist to move the cursor in the desired direction, and releases the pinch to end cursor motion. The user clutches and performs this repeatedly to reach the target.

5.5.1 Device and sensing: We used the same approach as Salemi Parizi et al. [77] to detect wrist angles via infrared (IR) sensors on a wristband. The detected wrist flexion/extension and radial/ulnar deviation at each timestamp were mapped to the wrist plane's x and y coordinates, respectively (see Figure 2b). With measurements at two consecutive timestamps, we derive the wrist velocity (V_w), and the detected wrist movement's angle (θ_s).

5.5.2 Determining cursor's moving direction: One important factor is the wearing position of the device. Ideally, the device should be placed exactly perpendicular to the body (Figure 2b). Then, moving the cursor along θ_s would match the user's intention. Yet, users wear the sensor slightly differently. The slight angular difference between the ideal and worn positions can cause the cursor to move in unintended angles (see Figure 2c).

We introduced an angular correction parameter θ_c, which is a value that directly adds to the sensed wrist angle such that $\theta_w = \theta_s + \theta_c$, where θ_w is the corrected angle, θ_s is the sensed angle, and θ_c is the correction parameter. After the correction, the cursor moves as intended θ_w. Figure 2d shows an effective correction. Thus, θ_c is an important design parameter that directly impacts TTD. Here, we define our relative transfer function:

$$C_x(t) = C_x(t-1) + f(V_w) \cdot \cos\theta_w, \quad C_y(t) = C_y(t-1) + f(V_w) \cdot \sin\theta_w \tag{8}$$

where $C_x(t)$, $C_y(t)$ are the cursor x, y at timestamp t. $f(V_w)$ determines the cursor velocity based on wrist velocity V_w.

5.5.3 Determining cursor's velocity: As in [65], our velocity function $f(V_w)$ is a sigmoid function, and consists of 3 parameters (c_1, c_2, c_3), which define the sigmoid curve properties (detailed in Table 3):

$$f(V_w) = \frac{1}{1 + e^{-c_1 \times (V_w - c_2)}} \cdot c_3, \tag{9}$$

In total, there are 4 design parameters to be optimized: the aforementioned angular correction, θ_c, and 3 parameters in the velocity function. From an optimization perspective, relative pointing is more complex than absolute. In addition to different users having different wrist rotation ranges, here tiny variations in wearability may lead to large sensing variations due to how IR sensors work. Additionally, since only the relative wrist motion matters, the user's arm is positioned downwards here (e.g., [60]), which enables relaxed use (see Figure 2a).

6 IMPLEMENTING TAF$^+$ WORKFLOW FOR OUR WRIST-BASED INTERACTIONS

In section 4, we introduced the details of each step in the TAF$^+$ workflow. Here, we report the process, data collection, and results of each step with our target interactions.

6.1 Step 1: Population modeling via a user data collection

For each interaction, participants performed the task while the parameters were changed every iteration based on multi-objective BO using Expected Hypervolume Improvement [19] and BoTorch [4][9]. The objective functions and design parameters were detailed in section 5. We ran 25 and 40 iterations for absolute and relative pointing, respectively. These numbers reflect the complexity of the tasks. These collected data, i.e., 25 and 40 pairs of (x, y), were used to construct the population models.

6.1.1 Procedure. We recruited 14 participants, 5 male, 8 female, and 1 non-binary, aged 22 – 57 ($median = 28.5$). None of them have any experience with VR. The data collection took 2 hours. Because we aim to analyze the two interactions independently, all participants went through relative pointing first and then absolute pointing, rather than mixing the order of the interactions. To familiarize participants with the task, we provided 10 "practice iterations", in which we uniformly sampled design instances from the entire design space. There was a 1-minute break every 10 iterations and a 5-minute break between the two interactions.

6.1.2 Deriving population models. We derived 14 population models for each interaction. **Appendix C** shows the plots of the hypervolume increase of two interactions at each iteration. On average, there were 4.8 ($s.d. = 0.78$) and 5.79 ($s.d. = 1.12$) Pareto-frontier settings for absolute pointing and relative pointing, respectively. This showed there are intrinsic trade-offs between the selected objective functions, and there is not a single optimal design that can maximize the three performance metrics.

6.2 Step 2: Objective weight α optimization based on user ratings

We determine the ideal values of the objective weights through optimization as described below.

6.2.1 Gathering user ratings. We compare the quality of the Pareto-optimal weight settings based on the participants' subjective ratings. For each participant, after they finished the 25 and 40 iterations in the previous step, we extracted all the Pareto-optimal parameter settings among these samples. Then, we asked the participants to perform pointing with these settings and rate them subjectively. The instruction was: "*For each of the following designs, please rate how much you agree with this statement: [This design is easy to use]. Please rate from 1 to 100; 1 stands for strongly disagree, and 100 means strongly agree.*" Participants interacted with each Pareto-optimal design twice in a randomized order. We took the average of the two ratings on the same parameter setting as its final rating. We normalized each user's ratings such that the lowest and highest ratings of a user would be 1 and 100. If the rating scale is scarce (e.g., only 5 or 7 levels), we may end up with many optimal weights that reach identical results when running the objective weight optimization. We therefore provided a 1-100 scale to gain the most granular information even if the users are not able to be exact in their assessments. To avoid a scenario where a user who rates all designs highly dominates the optimization process, we normalized each user's rating to a fixed range.

6.2.2 Results of objective weight optimization: We observed that 5 users see the 1-100 range as 10 levels (only rated at tens digits), 4 users see it as 20 levels (only rated at every five digits), and we further found 5 participants provided more fine-grained ratings (such as 73 or 92). This shows that the 1-100 scale allows each user to be flexible about the granularity they want to use for their scores. Following the procedures described in subsection 4.4, we sampled all the weight combinations to the first decimal and then performed the population-level objective weight optimization. The resulting optimal weight settings of [NCT, TAE, TTD] for absolute and relative pointing were [0.4, 0.3, 0.3] and [0.5, 0.3, 0.2], respectively. The resulting objective weights highlighted the consistent importance of speed (indicated by *NCT*) in pointing interactions.

6.3 Step 3: Decay hyperparameter optimization via simulations

We simulated the TAF$^+$ with different $< d_1, d_2 >$ values with the population models and observed which value pair results in the best performance. We set the $d1$ values to be [1, 2, 3, 4, 5, 6, 7, 8, 9] and the $d2$ values to be [0.1, 0.2, 0.3]. We created a list of $< d_1, d_2 >$ pairs of all the possible combinations. Additionally, we included two baselines in the simulation. The first was a standard single-objective Bayesian optimization where the objective was the weighted-sum objective. The second baseline was the TAF that handled multiple objectives using Equation 4 and no decay on the population model. For each $< d_1, d_2 >$ pair, we ran 14 simulations. In each simulation,

[9]Hyperparameter settings: We used a single-task GP with Matern 5/2 kernel. The single-task GP in our implementation infers a homoskedastic noise level of the observed data (https://botorch.org/docs/models). Similar to Chan et al. [15], we set $q = 1$. We further set 10 optimization restarts for the optimization of the acquisition function, 1024 restart candidates for the acquisition function optimization, and 512 Monte Carlo samples to approximate the acquisition function. A similar example can be found at https://botorch.org/tutorials/multi_objective_bo.

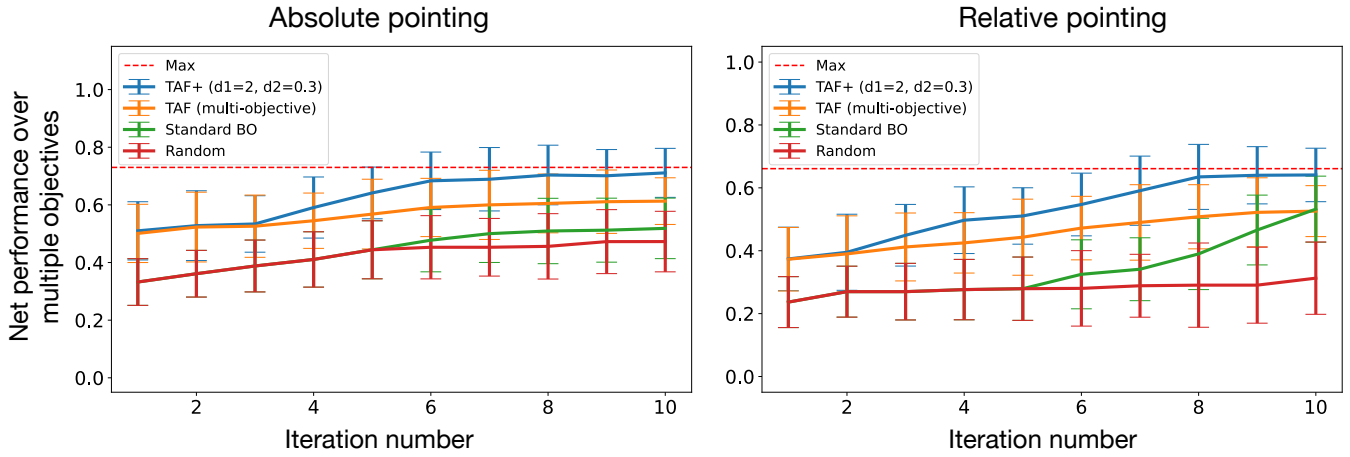

Figure 7: Result of the decay hyperparameter optimization. We simulated the user performance over 10 iterations with many combinations of hyperparameter values. Note that the TAF (multi-objective) condition handled multiple objectives with Equation 4. We derived the global maximum from a grid search of the GP prediction. The results showed that TAF$^+$ with proper decay setting led to the best performance. An extended version of this simulation is presented in Appendix D.

we singled out a population model and treated it as the new test user. We then excluded this model and utilized the remaining 13 population models to work with TAF$^+$. We set the optimization iteration to 10. Thus, each $< d_1, d_2 >$ pair resulted in 14 (users) $\times$10 (iterations) datapoints.

6.3.1 Results of the simulation. The simulation results are shown in Figure 7. With $d1 = 2, d2 = 0.3$, TAF$^+$ achieved the best performance for both interactions. Since there were too many combinations, we only show the performance of the best setting and two important baselines. TAF$^+$ with the optimal hyperparameter outperformed the standard BO. Additionally, TAF (multi-objective) had benefits in the early iterations but struggled to improve quickly, indicating directly deploying TAF without a mechanism to balance population models and the adaptation model may hinder adaptation efficiency. This issue would not exist if all the users had high similarities; the new user's optimal design would be highly aligned with the population. Yet, this assumption may not hold in certain scenarios, e.g., our interactions.

6.4 Summary

Through this workflow, we derived (1) 14 population models, (2) optimal OBJECTIVE WEIGHTS based on subjective ratings, and (3) optimal setting of hyperparameters ($d1$ and $d2$), which will be employed in the next adaptation phase.

7 ADAPTATION: EVALUATING META-BO ON NEW USERS

We evaluated the efficacy of our meta-BO (TAF$^+$) algorithm and workflow with 11 new participants for both pointing interactions. We evaluated against two baseline procedures: standard BO and manual calibration.

7.1 Experimental design

11 *entirely new* participants were recruited for the evaluation study: 6 males and 5 females, aged $23 - 42$ ($median = 29.5$). None of them have any experience pointing in VR or using VR. We conducted a within-subjects study with 2 independent variables: optimization procedure (meta-BO vs. standard BO vs. manual calibration) and iteration (10 iterations) for each of the pointing interactions. The procedures were counterbalanced using a Latin square.

7.2 Adaptation procedures (meta-BO, BO, and manual)

7.2.1 Meta-BO (TAF$^+$). We set up TAF$^+$ with 14 population models, optimized OBJECTIVE WEIGHTS, and decay hyperparameters as described earlier[10].

7.2.2 Standard BO. For the BO procedure, we had 5 initial random samplings and 5 optimization iterations. These values were determined from a 4-participant pilot test. To ensure a fair comparison, BO optimizes for the same weighted-sum objective obtained in Step 2, which TAF+ also optimizes for. Other hyperparameter settings are the same as in population modeling.

7.2.3 Manual calibration. Different from meta-BO and BO, the users adjust the weight settings based on their perception unrelated to the weighted-sum objective. For absolute pointing, an established way to calibrate for similar interactions is to ask the users to indicate their preferred operational range, and then map the maximum detected values to the boundary of the interface (E.g. [28, 29]). We followed the same, mapping the user's comfortable horizontal (yaw)

[10]The hyperparameter settings of the adaptation model followed the ones used in population modeling: a single-task GP with Matern 5/2 kernel and inferred noise levels. For computation efficiency, at each iteration, we generated 1024 parameter settings as candidates using a Sobol grid.

and vertical (pitch) forearm motion range to the scene boundaries along x, y.

For relative pointing, we first determined θ_c (correction parameter) for each user by using the radial-ulnar deviation of the user (corresponding to vertical cursor motion) since it has a smaller range than flexion-extension. The users were asked to perform radial-ulnar deviation to move the cursor back and forth between a top and a bottom target (Figure 6b). The difference between the recorded direction and the ideal vertical direction determined θ_c. Next, we calibrated the velocity function. Since c_3 controls the overall scale of the transfer function, we viewed this parameter as the most critical parameter. Users did 12 random target selections during which they were allowed to adjust c_3 by using a slider as many times as they wanted, similar to the sensitivity tuning of a computer mouse. As for c_1 and c_2, we used TAF$^+$ to propose the c_1 and c_2 values assuming there is no adaptation model; i.e., these are the best suggestions based on the population data. The values are 0.914 (c_1) and 1.438 (c_2). Unlike meta-BO and BO, manual calibration occurs before the data gathering. Once manual calibration was complete, the parameter values were fixed and did not change throughout the iterations.

7.3 Study procedure

Similar to step 1, participants performed relative pointing first, followed by absolute pointing. Instructions were the same as before. Different from step 1, each procedure only had 10 iterations. Since there were 3 procedures (meta-BO, BO, and Manual) and 10 iterations for each procedure, every pointing interaction had 30 iterations in total. After each procedure, we asked the participants to fill out the NASA-TLX questionnaire to assess their subjective workload. Afterward, we conducted an open-ended interview to understand the participants' experience.

7.4 Results

The weighted-sum performances at each iteration for both pointing interactions are shown in Figure 8.

7.4.1 Absolute pointing: There were two independent variables: the adaptation procedures (meta-BO, BO, Manual) and the iterations (1 - 10). A 2-way repeated-measures ANOVA with Greenhouse–Geisser correction showed no statistically significant interaction between the two variables ($F(1.651, 21.729) = 1.684$, $p = 0.194$). A simple main effects analysis found a significant difference between the procedures ($F(2, 30) = 1.652$, $p = 0.027$). Pairwise comparisons showed significant differences between meta-BO and BO, and between meta-BO and Manual (both $p < 0.05$), which indicated that meta-BO resulted in higher overall performances than the other two procedures. Overall, meta-BO enables performances that are on average 22.92% and 21.35% higher than BO and manual calibration across the 10 iterations. The simple main effects analysis also showed a significant difference between the iterations ($F(9, 100) = 2.171$, $p < 0.001$).

Another pairwise comparison between the iterations within each procedure found that for the meta-BO procedure, the performance did not significantly improve beyond iteration 6. On the other hand, BO still made significant improvements up to iteration 9. This indicates that meta-BO converges faster to optimal performance. We conducted a pairwise comparison between procedures for each iteration (Figure 8) which shows that meta-BO consistently outperforms BO and Manual in several iterations. The detailed numbers for Figure 8 are provided in **Appendix E**.

7.4.2 Relative pointing: A 2-way repeated-measures ANOVA showed no statistically significant interaction between procedure and iteration ($F(1.922, 29.403) = 1.623$, $p = 0.188$). A simple main effects analysis found a significant difference between the procedures ($F(2, 30) = 7.231$, $p = 0.005$). Pairwise comparisons showed significant differences between the meta-BO procedure and BO procedure ($p = 0.006$) and between the meta-BO procedure and the Manual procedure ($p = 0.033$), which indicated that meta-BO resulted in higher overall performances than the other two procedures. Averaging the performance of 10 iterations, meta-BO enables 25.43% and 13.60% better performances than BO and manual in relative pointing. A simple main effects analysis also showed a significant difference between the iterations ($F(9, 100) = 44.795$, $p < 0.001$). Pairwise comparisons between the iterations within each procedure showed that for meta-BO, the performance did not significantly improve after iteration 6. Meanwhile, BO improved up to iteration 7. Similar to absolute pointing, meta-BO is faster to converge to optimal performance. We conducted a pairwise comparison between procedures for each iteration (Figure 8) which shows meta-BO consistently outperforms BO and Manual in several iterations.

7.4.3 Other analyses: We found only one significant difference in the NASA-TLX questions: For absolute pointing, both Meta-BO and BO led to significantly lower frustration than the Manual procedure, indicating Meta-BO delivers better or comparable user experiences. This is mainly because the users needed to invest more effort in the calibration process but it did not lead to a better experience. From the interviews, we further learned that the participants' experience is highly influenced by the Index of Difficulty of targets, than by the transfer function. More detailed analyses on perceived workload and user experience are presented in **Appendix F**. We further analyzed the individual metrics derived from the three procedures for both interactions, and found that meta-BO generally led to comparable or significantly better performances than the baselines. Please refer to **Appendix G** for more details and the plots of the individual metrics. Finally, each user's performance is separately presented in **Appendix H**, and we visualized the objective function of absolute pointing in **Appendix I**.

7.5 Findings and discussion

Overall, we found that meta-BO allowed for significantly better performances than standard BO and manual calibration. For absolute pointing, standard BO started with a lower initial performance which is not surprising since it was not informed by prior population models. Standard BO further took more time to converge in both absolute and relative pointing. Even though meta-BO continued to improve until iteration 6, we can see that it reached near-peak performance by iteration 3, which translates to just 18 selections. Thus, meta-BO overcomes the slow start and slow convergence issues of standard BO for our tasks.

With manual calibration, since the participant calibrates it in the beginning according to their preference, the expectation would

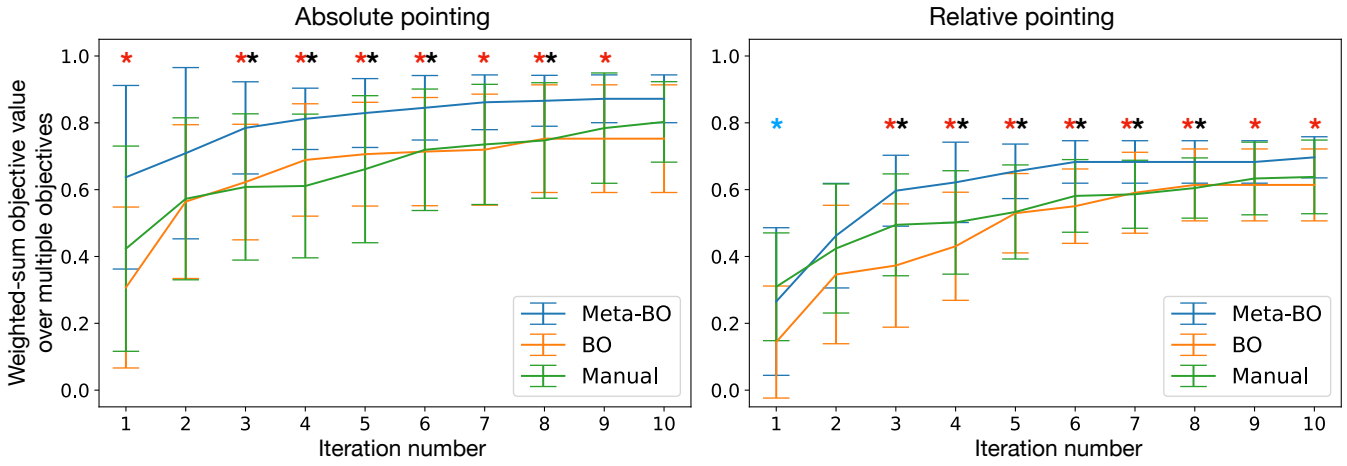

Figure 8: The performance (weighted-sum objective value over multiple objective functions) in 10 iterations. We have normalized all the objective functions into the range of $[0, 1]$. The error bar shows a 95% confidence interval. The red $*$ (meta-BO and BO), the black $*$ (meta-BO and Manual), and the blue $*$ (Manual and BO) signs denote a significant difference between two procedure at that iteration. Note that we showed "the best" performance reached from the beginning to each iteration. Since we are comparing the optimal performance given the same amount of iteration, this is the conventional way of showing and comparing the performance. Also note we have converted it to a maximization problem (see subsection 5.3). The detailed numbers of this figure are presented in Appendix E.

be that it will perform well at least in the beginning. While there are a few indications that the manual calibration procedure may have marginally better performance than BO, the difference is not significant. Thus, manual calibration does not guarantee an excellent result. This could be because users may not fine-tune to the extent of finding the optimal design. Even though the manual calibration parameters did not change, its performance improved over time, presumably because the participants were adapting over time. However, despite user adaptation, meta-BO consistently outperforms manual calibration for iterations 3-8. Given that meta-BO as an online procedure can match explicit manual calibration in the beginning and outperform it in intermediary trials, it is a viable candidate to replace explicit manual calibration procedures in real-world deployments.

8 DISCUSSION

In this work, we propose a novel HitL technique for *rapid, online* personalized parametric optimization for wrist-based input. It is known that adapting or optimizing transfer functions is challenging; wrist-based interaction adds further difficulties due to wearability and posture factors. We tackled the two most representative and distinct wrist-based input interactions. With just 14 users for population modeling, meta-BO outperforms the existing manual calibration and standard BO approaches for new users. This demonstrates the specific utility of meta-BO for interactions that benefit from personalized parametric settings. Meta-BO can eliminate dedicated calibration routines for wearable device interactions and help rapid attainment of optimal settings for each user. Given the results from two different pointing applications and a series of simulations and the fact that meta-BO does not have any overbearing assumptions (subsubsection 4.1.3), this work provides a meta-BO workflow that HCI researchers and practitioners can further apply to their applications and other problem contexts. We also showed how TAF$^+$ extended the TAF approach for HitL applications. This involved the derivation of optimal OBJECTIVE WEIGHTS based on users' subjective ratings and configuring decay hyperparameters through simulations to balance the population and adaptation models. The optimal weight derivation occurs after population modeling in our workflow, allowing higher flexibility to tune the OBJECTIVE WEIGHTS when needed. There are multiple open questions and limitations that future work can address.

Encountering new users: Meta-BO utilized the similarities between prior users and new users to converge to optimal settings faster. To account for a new user that is drastically different and the potential of negative transfer, we introduced decay parameters to let the optimization proactively rely on the observations from the new user. Our results show that with only 14 population models, meta-BO converges faster on new users on average. Further, we analyzed the individual users in **Appendix H** and found P5 (Table 14) as one example of a user who has drastically different optimal parameters from others. As we see in P5's performance plotted in Figure 25, meta-BO is still able to converge to a performance comparable to the baselines. It also shows the benefits of having a diverse initial user set. Of course, when a user behaves even more extreme, such as wearing the device completely wrongly or their optimal parameter setting is beyond the parameter range, TAF$^+$ can not adapt for such cases. To address this challenge, future work could consider developing mechanisms to diagnose a new user's performance in real-time and switch to standard BO or manual optimization.

Enhancing the efficiency, scalability, and determining the OBJECTVIE WEIGHTS*:* To enable more efficient calibration of wrist-based interactions, it is worth exploring more advanced normalization techniques to determine the quality of a setting with fewer selections. In addition, although TAF/TAF$^+$ is relatively lightweight, its computation cost is linearly increased by the number of models, introducing difficulties when scaling up the population models. This issue could be mitigated by calculating the *EI* and *PEIs* in parallel. Further, the current approach to deriving optimal OBJECTIVE WEIGHTS based on subjective ratings of the population may not be suitable for all new users. Future research could investigate deriving the individual user's optimal OBJECTIVE WEIGHTS by user feedback during adaptation; preferential Bayesian optimization [30] may be incorporated in this direction.

Potential co-adaptation: In the evaluation of TAF$^+$, there were only a few observations in each adaptation procedure, and each parameter setting was evaluated only once, making it hard to detect user learning. Future work should consider developing methods that take the user to revisit certain parameter settings for better-estimating user learning. Also, more advanced computational methods are also needed to infer the user's learning.

Multi-objective TAF:. Meta-BO is an emerging topic with many open research questions and opportunities. The field of HCI can benefit highly from it. In addition to applying TAF$^+$ to other applications, future research can investigate other meta-BO methods for HitL optimization. One potential direction is extending TAF for multi-objective tasks by changing the base acquisition function to Expected Hypervolume Increase (EHVI). We ran a simulation with this approach using the population models, and the results showed that TAF outperformed the standard multi-objective BO, as plotted in **Appendix J**. However, multi-objective TAF would require the end-users to engage with extreme designs that heavily prioritize one objective while neglecting others. Further, one would need a heuristic to select one design from the Pareto-frontier which may or may not be the one preferred by the user. Alternatively, the user will need to manually determine one final setting among the Pareto-frontier through trials, which will introduce more effort to the users. Finally, the time required for computing the multi-objective TAF to yield the next parameter setting during adaptation is massive for each iteration because the computation complexity is N (the total number of models) multiplied by the complexity for computing EHVI from a model. Thus, it would be unsuitable for online adaptations which require a fast turn-around time.

9 CONCLUSION

In this paper, we present an online, fast-converging parametric optimization procedure through meta-Bayesian optimization. We introduce a novel meta-BO algorithm, TAF$^+$, and a tailored workflow to meet the unique requirements of human-in-the-loop problems. We apply TAF$^+$ and its workflow to two distinct wrist-based interactions. The positive result and the outcome of each step showcase the effectiveness and efficiency of meta-BO compared to conventional calibration procedures and state-of-the-art BO. Calibration is a common practice among HCI practitioners and researchers, typically created by developers or designers. Crafting an effective calibration procedure is challenging and often specific to a particular device or interaction. Moreover, our study indicated that manual calibration does not always yield optimal results. Our success in wrist-based pointing demonstrates that meta-BO holds significant potential as a general calibration method across various interactions and devices, eliminating the difficulties associated with designing calibration procedures and achieving a better user experience. We encourage future research to explore the application of meta-BO for optimizing parameter settings across diverse applications, both within HCI and beyond. We anticipate this work will pave the way for more personalized and adaptive user interfaces.

ACKNOWLEDGMENTS

We would like to thank Dr. Kashyap Todi for his suggestions on interaction metrics and discussions throughout the project. We extend our gratitude to our colleagues in Reality Labs Research for their unwavering support. Additionally, we wholeheartedly thank Dr. Hee-Seung Moon, Dr. John J. Dudley, and Prof. Antti Oulasvirta for their valuable comments, which significantly enhanced the quality of the paper.

REFERENCES

[1] Karan Ahuja, Sven Mayer, Mayank Goel, and Chris Harrison. 2021. Pose-on-the-Go: Approximating User Pose with Smartphone Sensor Fusion and Inverse Kinematics. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 9, 12 pages. https://doi.org/10.1145/3411764.3445582

[2] Ahmed Ayman, Omneya Attalah, and Heba Shaban. 2019. An Efficient Human Activity Recognition Framework Based on Wearable IMU Wrist Sensors. In *2019 IEEE International Conference on Imaging Systems and Techniques (IST)*. 1–5. https://doi.org/10.1109/IST48021.2019.9010115

[3] Tianyi Bai, Yang Li, Yu Shen, Xinyi Zhang, Wentao Zhang, and Bin Cui. 2023. Transfer Learning for Bayesian Optimization: A Survey. *arXiv preprint arXiv:2302.05927* (2023).

[4] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. 2020. BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems* 33 (2020), 21524–21538.

[5] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. 2013. Collaborative Hyperparameter Tuning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (Atlanta, GA, USA) *(ICML'13)*. JMLR.org, II–199–II–207.

[6] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. 1990. *Learning a synaptic learning rule.* Citeseer.

[7] Edwin V Bonilla, Kian Chai, and Christopher Williams. 2007. Multi-task Gaussian process prediction. *Advances in neural information processing systems* 20 (2007).

[8] Ali Borji and Laurent Itti. 2013. Bayesian optimization explains human active search. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/a3f390d88e4c41f2747bfa2f1b5f87db-Paper.pdf

[9] Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010. A Bayesian Interactive Optimization Approach to Procedural Animation Design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar, DEU, 103–112.

[10] Grigore C Burdea and Philippe Coiffet. 2003. *Virtual reality technology.* John Wiley & Sons.

[11] Géry Casiez and Nicolas Roussel. 2011. No More Bricolage! Methods and Tools to Characterize, Replicate and Compare Pointing Transfer Functions. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 603–614. https://doi.org/10.1145/2047196.2047276

[12] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1€ Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems. In *CHI'12, the 30th Conference on Human Factors in Computing Systems*. ACM, Austin, United States, 2527–2530. https://doi.org/10.1145/2207676.2208639

[13] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *Human–Computer Interaction* 23, 3 (2008), 215–250. https://doi.org/10.1080/07370020802278163 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/07370020802278163

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

[15] Liwei Chan, Yi-Chi Liao, George B Mo, John J Dudley, Chun-Lien Cheng, Per Ola Kristensson, and Antti Oulasvirta. 2022. Investigating Positive and Negative Qualities of Human-in-the-Loop Optimization for Designing Interaction Techniques. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 112, 14 pages. https://doi.org/10.1145/3491102.3501850

[16] Siyuan Chen, Julien Epps, Natalie Ruiz, and Fang Chen. 2011. Eye Activity as a Measure of Human Mental Effort in HCI. In *Proceedings of the 16th International Conference on Intelligent User Interfaces* (Palo Alto, CA, USA) *(IUI '11)*. Association for Computing Machinery, New York, NY, USA, 315–318. https://doi.org/10.1145/1943403.1943454

[17] Chia-Hsing Chiu, Yuki Koyama, Yu-Chi Lai, Takeo Igarashi, and Yonghao Yue. 2020. Human-in-the-Loop Differential Subspace Search in High-Dimensional Latent Space. *ACM Trans. Graph.* 39, 4, Article 85 (aug 2020), 15 pages. https://doi.org/10.1145/3386569.3392409

[18] Zhongxiang Dai, Yizhou Chen, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. 2022. On provably robust meta-Bayesian optimization. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 180)*, James Cussens and Kun Zhang (Eds.). PMLR, 475–485. https://proceedings.mlr.press/v180/dai22a.html

[19] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. 2021. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems* 34 (2021), 2187–2200.

[20] Laura Dipietro, Angelo M. Sabatini, and Paolo Dario. 2008. A Survey of Glove-Based Systems and Their Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 4 (2008), 461–482. https://doi.org/10.1109/TSMCC.2008.923862

[21] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* (2016).

[22] Michael Emmerich and Jan-willem Klinkenberg. 2008. The computation of the expected improvement in dominated hypervolume of Pareto front approximations. *Rapport technique, Leiden University* 34 (2008), 7–3.

[23] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. 2018. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML Workshop at ICML*, Vol. 7.

[24] Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. 2018. Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219* (2018).

[25] Richard S Figliola and Donald E Beasley. 2020. *Theory and design for mechanical measurements*. John Wiley & Sons.

[26] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[27] Peter I Frazier. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).

[28] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WristText: One-Handed Text Entry on Smartwatch Using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3173755

[29] Jun Gong, Xing-Dong Yang, and Pourang Irani. 2016. WristWhirl: One-Handed Continuous Smartwatch Input Using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 861–872. https://doi.org/10.1145/2984511.2984563

[30] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. 2017. Preferential bayesian optimization. In *International Conference on Machine Learning*. PMLR, 1282–1291.

[31] Deepak Gopinath, Siddarth Jain, and Brenna D Argall. 2016. Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters* 2, 1 (2016), 247–254.

[32] Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. 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, 59–70. https://doi.org/10.1145/2858036.2858052

[33] Faizan Haque, Mathieu Nancel, and Daniel Vogel. 2015. Myopoint: Pointing and Clicking Using Forearm Mounted Electromyography and Inertial Motion Sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3653–3656. https://doi.org/10.1145/2702123.2702133

[34] Juan David Hincapié-Ramos, Kasim Özacar, Pourang P. Irani, and Yoshifumi Kitamura. 2016. GyroWand: An Approach to IMU-Based Raycasting for Augmented Reality. *IEEE Computer Graphics and Applications* 36, 2 (2016), 90–96. https://doi.org/10.1109/MCG.2016.21

[35] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44, 9 (2021), 5149–5169.

[36] Bing-Jing Hsieh, Ping-Chun Hsieh, and Xi Liu. 2021. Reinforced few-shot acquisition function learning for bayesian optimization. *Advances in Neural Information Processing Systems* 34 (2021), 7718–7731.

[37] Carl Hvarfner, Danny Stoll, Artur Souza, Marius Lindauer, Frank Hutter, and Luigi Nardi. 2022. πBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. *arXiv preprint arXiv:2204.11051* (2022).

[38] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13 (1998), 455–492.

[39] Florian Kadner, Yannik Keller, and Constantin Rothkopf. 2021. Adaptifont: Increasing individuals' reading speed with a generative font model and bayesian optimization. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–11.

[40] Frederic Kerber, Pascal Lessel, and Antonio Krüger. 2015. Same-Side Hand Interactions with Arm-Placed Devices Using EMG. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 1367–1372. https://doi.org/10.1145/2702613.2732895

[41] Mohammad M. Khajah, Brett D. Roads, Robert V. Lindsey, Yun-En Liu, and Michael C. Mozer. 2016. Designing Engaging Games Using Bayesian Optimization. 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, 5571–5582. https://doi.org/10.1145/2858036.2858253

[42] Wolf Kienzle, Eric Whitmire, Chris Rittaler, and Hrvoje Benko. 2021. ElectroRing: Subtle Pinch and Touch Detection with a Ring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 3, 12 pages. https://doi.org/10.1145/3411764.3445094

[43] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: Freehand 3D Interactions Anywhere Using a Wrist-Worn Gloveless Sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 167–176. https://doi.org/10.1145/2380116.2380139

[44] Jiwan Kim and Ian Oakley. 2022. SonarID: Using Sonar to Identify Fingers on a Smartwatch. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 287, 10 pages. https://doi.org/10.1145/3491102.3501935

[45] Myunghee Kim, Ye Ding, Philippe Malcolm, Jozefien Speeckaert, Christoper J. Siviy, Conor J. Walsh, and Scott Kuindersma. 2017. Human-in-the-loop Bayesian optimization of wearable device parameters. *PLOS ONE* 12, 9 (09 2017), 1–15. https://doi.org/10.1371/journal.pone.0184054

[46] Alice Y Kolb and David A Kolb. 2009. The learning way: Meta-cognitive aspects of experiential learning. *Simulation & gaming* 40, 3 (2009), 297–327.

[47] Yuki Koyama and Masataka Goto. 2022. BO as Assistant: Using Bayesian Optimization for Asynchronously Generating Design Suggestions. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 77, 14 pages. https://doi.org/10.1145/3526113.3545664

[48] Yuki Koyama, Issei Sato, and Masataka Goto. 2020. Sequential Gallery for Interactive Visual Design Optimization. *ACM Trans. Graph.* 39, 4, Article 88 (aug 2020), 12 pages. https://doi.org/10.1145/3386569.3392444

[49] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.

[50] Yeara Kozlov and Tino Weinkauf. 2013. Persistence1D. https://github.com/yeara/Persistence1D.

[51] Ananda Sankar Kundu, Oishee Mazumder, Prasanna Kumar Lenka, and Subhasis Bhaumik. 2018. Hand gesture recognition based omnidirectional wheelchair control using IMU and EMG sensors. *Journal of Intelligent & Robotic Systems* 91 (2018), 529–541.

[52] Byungjoo Lee, Mathieu Nancel, Sunjun Kim, and Antti Oulasvirta. 2020. AutoGain: Gain Function Adaptation with Submovement Efficiency Optimization. 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, 1–12. https://doi.org/10.1145/3313831.3376244

[53] Lik Hang Lee, Ngo Yan Yeung, Tristan Braud, Tong Li, Xiang Su, and Pan Hui. 2020. Force9: Force-Assisted Miniature Keyboard on Smart Wearables. In *Proceedings of the 2020 International Conference on Multimodal Interaction* (Virtual Event, Netherlands) *(ICMI '20)*. Association for Computing Machinery, New York, NY, USA, 232–241. https://doi.org/10.1145/3382507.3418827

[54] Wendi Li, Ting Wang, and Wing W. Y. Ng. 2021. Population-Based Hyperparameter Tuning With Multitask Collaboration. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–13. https://doi.org/10.1109/TNNLS.2021.3130896

[55] Yang Li, Yu Shen, Huaijun Jiang, Wentao Zhang, Zhi Yang, Ce Zhang, and Bin Cui. 2022. TransBO: Hyperparameter Optimization via Two-Phase Transfer Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 956–966.

[56] Yi-Chi Liao, John J. Dudley, George B. Mo, Chun-Lien Cheng, Liwei Chan, Antti Oulasvirta, and Per Ola Kristensson. 2023. Interaction Design With Multi-objective Bayesian Optimization. *IEEE Pervasive Computing* (2023), 1–10. https://doi.org/10.1109/MPRV.2022.3230597

[57] Yi-Chi Liao, Sunjun Kim, Byungjoo Lee, and Antti Oulasvirta. 2020. Button Simulation and Design via FDVV Models. 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, 1–14. https://doi.org/10.1145/3313831.3376262

[58] Kieran Little, Chris W. Antuvan, Michele Xiloyannis, Bernardo A.P.S. de Noronha, Yongtae G. Kim, Lorenzo Masia, and Dino Accoto. 2019. IMU-based assistance modulation in upper limb soft wearable exosuits. In *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*. 1197–1202. https://doi.org/10.1109/ICORR.2019.8779362

[59] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675. https://doi.org/10.1016/j.pmcj.2009.07.007 PerCom 2009.

[60] Mingyu Liu, Mathieu Nancel, and Daniel Vogel. 2015. Gunslinger: Subtle Arms-down Mid-Air Interaction *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 63–71. https://doi.org/10.1145/2807442.2807489

[61] I. Scott MacKenzie. 1992. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Hum.-Comput. Interact.* 7, 1 (mar 1992), 91–139. https://doi.org/10.1207/s15327051hci0701_3

[62] J. Mockus. 1975. On the Bayes Methods for Seeking the Extremal Point. *IFAC Proceedings Volumes* 8, 1, Part 1 (1975), 428–431. https://doi.org/10.1016/S1474-6670(17)67769-3 6th IFAC World Congress (IFAC 1975) - Part 1: Theory, Boston/Cambridge, MA, USA, August 24-30, 1975.

[63] Alessandra Moschetti, Laura Fiorini, Dario Esposito, Paolo Dario, and Filippo Cavallo. 2016. Recognition of Daily Gestures with Wearable Inertial Rings and Bracelets. *Sensors* 16, 8 (2016). https://doi.org/10.3390/s16081341

[64] Alessandra Moschetti, Laura Fiorini, Dario Esposito, Paolo Dario, and Filippo Cavallo. 2017. Daily activity recognition with inertial ring and bracelet: An unsupervised approach. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 3250–3255. https://doi.org/10.1109/ICRA.2017.7989370

[65] Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015. Mid-Air Pointing on Ultra-Walls. *ACM Trans. Comput.-Hum. Interact.* 22, 5, Article 21 (aug 2015), 62 pages. https://doi.org/10.1145/2766448

[66] Mathieu Nancel, Daniel Vogel, and Edward Lank. 2015. Clutching Is Not (Necessarily) the Enemy. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 4199–4202. https://doi.org/10.1145/2702123.2702134

[67] Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2, 3 (2018), 4.

[68] Jens Brehm Nielsen, Jakob Nielsen, Bjørn Sand Jensen, and Jan Larsen. 2013. Hearing aid personalization. (2013).

[69] Jens Brehm Bagger Nielsen, Jakob Nielsen, and Jan Larsen. 2015. Perception-Based Personalization of Hearing Aids Using Gaussian Processes and Active Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 1 (2015), 162–173. https://doi.org/10.1109/TASLP.2014.2377581

[70] Michael A Osborne, Roman Garnett, and Stephen J Roberts. 2009. Gaussian processes for global optimization. In *3rd international conference on learning and intelligent optimization (LION3)*. Springer-Verlag Berlin, Germany, 1–15.

[71] Kasım Özacar, Juan David Hincapié-Ramos, Kazuki Takashima, and Yoshifumi Kitamura. 2017. 3d selection techniques for mobile augmented reality head-mounted displays. *Interacting with Computers* 29, 4 (2017), 579–591.

[72] J.K. Perng, B. Fisher, S. Hollar, and K.S.J. Pister. 1999. Acceleration sensing glove (ASG). In *Digest of Papers. Third International Symposium on Wearable Computers*. 178–180. https://doi.org/10.1109/ISWC.1999.806717

[73] Ken Pfeuffer, Melodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. 2013. Pursuit Calibration: Making Gaze Calibration Less Tedious and More Flexible. 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, 261–270. https://doi.org/10.1145/2501988.2501998

[74] Iman Prayudi and Doik Kim. 2012. Design and implementation of IMU-based human arm motion capture system. In *2012 IEEE International Conference on Mechatronics and Automation*. 670–675. https://doi.org/10.1109/ICMA.2012.6283221

[75] J. Rekimoto. 2001. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *Proceedings Fifth International Symposium on Wearable Computers*. 21–27. https://doi.org/10.1109/ISWC.2001.962092

[76] Yi Ren and Panos Y Papalambros. 2011. A design preference elicitation query as an optimization process. *Journal of Mechanical Design* 133, 11 (2011).

[77] Farshid Salemi Parizi, Wolf Kienzle, Eric Whitmire, Aakar Gupta, and Hrvoje Benko. 2021. RotoWrist: Continuous Infrared Wrist Angle Tracking Using a Wristband. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology* (Osaka, Japan) *(VRST '21)*. Association for Computing Machinery, New York, NY, USA, Article 26, 11 pages. https://doi.org/10.1145/3489849.3489886

[78] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-Learning with Memory-Augmented Neural Networks. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1842–1850. https://proceedings.mlr.press/v48/santoro16.html

[79] T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A. Landay. 2009. Enabling Always-Available Input with Muscle-Computer Interfaces. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 167–176. https://doi.org/10.1145/1622176.1622208

[80] Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2016. Scalable Hyperparameter Optimization with Products of Gaussian Process Experts. In *Machine Learning and Knowledge Discovery in Databases*, Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken (Eds.). Springer International Publishing, Cham, 33–48.

[81] Jürgen Schmidhuber. 1992. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation* 4, 1 (1992), 131–139. https://doi.org/10.1162/neco.1992.4.1.131

[82] Nicolas Schweighofer and Kenji Doya. 2003. Meta-learning in Reinforcement Learning. *Neural Networks* 16, 1 (2003), 5–9. https://doi.org/10.1016/S0893-6080(02)00228-9

[83] Matthias Seeger. 2004. Gaussian processes for machine learning. *International journal of neural systems* 14, 02 (2004), 69–106.

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

[85] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175. https://doi.org/10.1109/JPROC.2015.2494218

[86] Ahmed Sharshar, Ahmed Fayez, Yasser Ashraf, and Walid Gomaa. 2021. Activity With Gender Recognition Using Accelerometer and Gyroscope. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. 1–7. https://doi.org/10.1109/IMCOM51814.2021.9377388

[87] Junxiao Shen, Jinghui Hu, John J. Dudley, and Per Ola Kristensson. 2022. Personalization of a Mid-Air Gesture Keyboard using Multi-Objective Bayesian Optimization. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 702–710. https://doi.org/10.1109/ISMAR55827.2022.00088

[88] Edward Snelson and Zoubin Ghahramani. 2005. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems* 18 (2005).

[89] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat Prabhat, and Ryan P. Adams. 2015. Scalable Bayesian Optimization Using Deep Neural Networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (Lille, France) *(ICML'15)*. JMLR.org, 2171–2180.

[90] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. *Advances in neural information processing systems* 29 (2016).

[91] Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-Task Bayesian Optimization. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf

[92] Sebastian Thrun and Lorien Pratt (Eds.). 1998. *Learning to Learn*. Kluwer Academic Publishers, USA.

[93] Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. 2019. Meta-learning acquisition functions

for transfer learning in bayesian optimization. *arXiv preprint arXiv:1904.02642* (2019).

[94] Jane X Wang. 2021. Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences* 38 (2021), 90–95. https://doi.org/10.1016/j.cobeha.2021.01.002 Computational cognitive neuroscience.

[95] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. 2022. Recent Advances in Bayesian Optimization. *arXiv preprint arXiv:2206.03301* (2022).

[96] Yuntao Wang, Jiexin Ding, Ishan Chatterjee, Farshid Salemi Parizi, Yuzhou Zhuang, Yukang Yan, Shwetak Patel, and Yuanchun Shi. 2022. FaceOri: Tracking Head Position and Orientation Using Ultrasonic Ranging on Earphones. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 290, 12 pages. https://doi.org/10.1145/3491102.3517698

[97] Yuyang Wang and Roni Khardon. 2012. Sparse Gaussian processes for multi-task learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part I 23*. Springer, 711–727.

[98] James Wilson, Frank Hutter, and Marc Deisenroth. 2018. Maximizing acquisition functions for Bayesian optimization. *Advances in neural information processing systems* 31 (2018).

[99] Martin Wistuba and Josif Grabocka. 2021. Few-shot Bayesian optimization with deep kernel surrogates. *arXiv preprint arXiv:2101.07667* (2021).

[100] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. 2018. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning* 107, 1 (2018), 43–78.

[101] Jacob O. Wobbrock, James Fogarty, Shih-Yen (Sean) Liu, Shunichi Kimuro, and Susumu Harada. 2009. The Angle Mouse: Target-Agnostic Dynamic Gain Adjustment Based on Angular Deviation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1401–1410. https://doi.org/10.1145/1518701.1518912

[102] Charence Wong, Zhi-Qiang Zhang, Benny Lo, and Guang-Zhong Yang. 2015. Wearable Sensing for Solid Biomechanics: A Review. *IEEE Sensors Journal* 15, 5 (2015), 2747–2760. https://doi.org/10.1109/JSEN.2015.2393883

[103] Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya Parameswaran. 2018. Accelerating human-in-the-loop machine learning: Challenges and opportunities. In *Proceedings of the second workshop on data management for end-to-end machine learning*. 1–4.

[104] Kenta Yamamoto, Yuki Koyama, and Yoichi Ochiai. 2022. Photographic Lighting Design with Photographer-in-the-Loop Bayesian Optimization. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 92, 11 pages. https://doi.org/10.1145/3526113.3545690

[105] Hui-Shyong Yeo, Wenxin Feng, and Michael Xuelin Huang. 2020. WATouCH: Enabling Direct Input on Non-Touchscreen Using Smartwatch's Photoplethysmogram and IMU Sensor Fusion. 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, 1–10. https://doi.org/10.1145/3313831.3376198

[106] Hui-Shyong Yeo, Erwin Wu, Juyoung Lee, Aaron Quigley, and Hideki Koike. 2019. Opisthenar: Hand Poses and Finger Tapping Recognition by Observing Back of Hand Using Embedded Wrist Camera. 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, 963–971. https://doi.org/10.1145/3332165.3347867

[107] Dani Yogatama and Gideon Mann. 2014. Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics*. PMLR, 1077–1085.

[108] Jaesik Yun, Youn kyung Lim, Kee-Eung Kim, and Seokyoung Song. 2015. Interactivity Crafter: An Interactive Input-Output Transfer Function Design Tool for Interaction Designers. *Archives of Design Research* 28 (2015), 21–37. https://api.semanticscholar.org/CorpusID:64149473

[109] Yang Zhang and Chris Harrison. 2015. Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 167–173. https://doi.org/10.1145/2807442.2807480

A DETAILS OF OBJECTIVE WEIGHTS OPTIMIZATION

A.1 A demonstrative example of OBJECTIVE WEIGHTS optimization with two users

To explain the need for the across-user objective weight optimization approach in step 2 of our workflow, we provide an example where two users are involved. User A's and User B's Pareto-optimal performances are presented in Table 4 and Table 5, respectively. Only by examining User A's profile (Table 4), design B has the highest user rating, and its third objective has the highest value. Therefore, setting weights as $[0.1, 0.1, 0.8]$ is a straightforward configuration that leads to the highest user rating. On the other hand, examining only User B's profile (Table 5), design A has the highest user rating, and its first objective has the highest value, so one could conclude with weights $[0.8, 0.1, 0.1]$. However, looking at the group level, setting weights as $[0.1, 0.1, 0.8]$ or $[0.8, 0.1, 0.1]$ to both users at the same time would result in the other user ending up with a suboptimal design (i.e., the design not associated with the highest user rating). On the contrary, with an appropriate search, setting OBJECTIVE WEIGHTS as $[0.4, 0.2, 0.2]$ would allow both users to end up having the designs with the highest ratings. With more users taken into consideration, it becomes increasingly challenging to directly see which OBJECTIVE WEIGHT configuration is optimal for the group. Hence, a principled grid search is an easier and more principled approach.

A.2 Algorithm for OBJECTIVE WEIGHT Optimization

Altorighm 1 presents the details of OBJECTIVE WEIGHT optimization.

Pareto-optimal designs/settings	Original objective values	Weighted-sum objective value (weights=[0.8, 0.1, 0.1])	Weighted-sum objective value (weights=[0.1, 0.1, 0.8])	*Weighted-sum objective value (weights=[0.4, 0.2, 0.4])*	User rating
A	[7, 2, 5]	**6.4**	4.9	4.8	20
B	[5, 4, 8]	5.2	**7.3**	*6.4*	**100**
C	[3, 9, 2]	3.5	2.8	3.8	1

Table 4: The table shows a set of Pareto-optimal designs in a three-objective optimization problem of an example User A. Intuitively, setting the OBJECTIVE WEIGHTS as [0.1, 0.1, 0.8] would easily end up having design B, which has the highest user rating (100). However, when taking the second user (presented in Table 5) into consideration, [0,1, 0.1, 0.8] would bring the suboptimal design (i.e., not associated with the highest rating) to the other. However, setting OBJECTIVE WEIGHTS as [0.4, 0.2, 0.4] leads to the design of the highest ratings for both users.

Pareto-optimal designs/settings	Original objective values	Weighted-sum objective value (weights=[0.8, 0.1, 0.1])	Weighted-sum objective value (weights=[0.1, 0.1, 0.8])	*Weighted-sum objective value (weights=[0.4, 0.2, 0.4])*	User rating
A	[8, 3, 5]	**7.2**	5.1	*5.8*	**100**
B	[2, 1, 7]	2.4	**5.9**	3.8	50
C	[4, 4, 3]	3.9	3.2	3.6	1

Table 5: The table shows a set of Pareto-optimal designs in a three-objective optimization problem with an example User B. Intuitively, setting the OBJECTIVE WEIGHTS as [0.8, 0.1, 0.1] would end up having design A, which has the highest user rating (100). However, this weight setting is not optimal for the other user (presented in Table 4). While not intuitive, setting OBJECTIVE WEIGHTS as [0.4, 0.2, 0.4] leads to the designs with the highest ratings for both.

Algorithm 1 Optimize OBJECTIVE WEIGHT configuration based on user ratings.

Inputs:
$j \in [1, J]$ for there are J users in total; j stands for the j-th user among all users.
K_j is the total number of Pareto-optimal designs for the j-th user. For instance, the 5th user tries out 10 parameter settings, and 3 of them result in Pareto-optimal performance; then $K_5 = 3$.
$k \in [1, K_j]$ denotes the j-th user's k-th Pareto-optimal design.
$PO_j^k[1..L]$ is the j-th user's k-th Pareto-optimal objective value set, which contains L objective functions.
R_j^k is the j-th user's rating for the k-th Pareto-optimal designs.
Outputs:
$optimal\_weight[1..L]$, an optimal weight setting for L objectives.
Initialize:
$weights \leftarrow$ All possible non-zero weight combinations to the first decimal point.
(e.g., $weights[0] = [0.8, 0.1, 0.1]$, $weights[1] = [0.7, 0.2.0.1]$ ...)
$scores \leftarrow []$
for $j \in [1, J]$ **do** ▷ Normalize user ratings.
  $R_{max} \leftarrow \max(R_j)$ ▷ Get the maximum rating of user j.
  $R_{min} \leftarrow \min(R_j)$ ▷ Get the minimum rating of user j.
  for $k \in [1, K_j]$ **do**
    $R_j^k \leftarrow 1 + 99 \cdot \frac{R_j^k - R_{min}}{R_{max} - R_{min}}$ ▷ Normalize the user rating to be in the range of [1, 100].
  end for
end for
for $weight \in weights$ **do** ▷ Start the actual optimization. Loop over all the weight settings.
  $score\_of\_this\_weight = 0$
  for $j \in J$ **do** ▷ Loop over every user.
    $index \leftarrow \underset{k \in K_j}{\text{argmax}}(\sum_{m=1}^{L} weight[m] \cdot PO_j^k[m])$ ▷ Get the index of the highest weighted sum.
    $score\_of\_this\_weight \leftarrow score\_of\_this\_weight + R_j^{index}$ ▷ Add the rating of the highest value to the score.
  end for
  $scores$.append($score\_of\_this\_weight$)
end for
$final\_index \leftarrow argmax(scores)$ ▷ Get the index of the weight setting with the highest score.
$optimal\_weight \leftarrow weights[final\_index]$ ▷ Get the optimal weight setting.
return $optimal\_weight$ ▷ Return the optimal weight setting.

Parameters	Corresponding Objective Functions	Range
x_1	Sphere 1, center = (0.55, 0.4)	[0, 1]
x_2	Sphere 1, center = (0.55, 0.4) & Sphere 2 (center = (0.6, 0.45)	[0, 1]
x_3	Sphere 2, center = (0.6, 0.45) & Sphere 3, center = (0.65, 0.35)	[0, 1]
x_4	Sphere 3, center = (0.65, 0.35)	[0, 1]

Table 6: The parameters of the *base function* used in our Simulation 1 and Simulation 2. Here we present the corresponding objective functions of each parameter and the parameters' ranges.

B EVALUATING TAF$^+$'S VIABILITY VIA SIMULATIONS ON SYNTHETIC FUNCTIONS

We evaluate the performance of our proposed TAF$^+$ in *multi-objective* tasks through simulations with commonly used testing functions. For the evaluation of TAF in single-objective problems, please refer to Wistuba et al. [100] and Volpp et al. [93]. Here, we aim to answer four goals through these simulations:

- Simulation 1: Evaluating the robustness of TAF$^+$ under diverse OBJECTIVE WEIGHT configurations.
- Simulation 2: Validating the effectiveness of TAF$^+$ under varying levels of user similarity.
- Simulation 3: Investigating the generalizability of TAF$^+$ across different testing functions.
- Simulation 4: Investigating the performances of TAF$^+$ with different numbers of population models.

B.1 Base function for simulation 1 and simulation 2

In Simulations 1 and 2, we utilize the same *base function*, which has four design parameters and three objective functions. The number of parameters and functions aligns with our relative pointing interaction. Specifically, we have adopted the 2-dimensional Sphere function[11] as our choice for objective functions. To elaborate further, our Sphere function is mathematically defined as follows: $y = 1 - \sum_{i=1}^{2}(\dot{x}_i - X_i)^2 \times \gamma$. Here, $\dot{x}_i$ represents the selected value of a design parameter, and X_i denotes the center of the square function. This Sphere function has its maximum objective value ($y = 1$) when the selected $\dot{x}$ is right at the center position; i.e., $(\dot{x}_1, \dot{x}_2) = (X_1, X_2)$. As the parameter values deviate from this central point, the objective values gradually decrease. We incorporate an additional coefficient γ, set to be 8, to accelerate the rate of decay in the function. We chose the Sphere function for two reasons. Firstly, it effectively simulates the real-world scenario where a user's performance gradually decreases as they deviate from the optimal design parameter setting. Secondly, the Sphere function is widely recognized and used for similar optimization evaluations.

We then utilize the Sphere function to construct the *base function*, which contains four parameters (x_1, x_2, x_3, x_4) and 3 objective functions (y_1, y_2, y_3). Details are provided in Table 6. We first create three distinct Sphere functions, each having a unique center position (X_1, X_2). Consequently, there is no single parameter setting that maximizes all three Sphere functions simultaneously. Furthermore, the first two parameters in the *base function* (x_1, x_2) contribute to the first Sphere function, (x_2, x_3) contribute to the second Sphere function, and (x_3, x_4) contribute to the third Sphere function. Parameters x_2 and x_3 are shared by **two** Sphere functions. This decision was made to introduce trade-off scenarios: there exists no single x_2 or x_3 value that can optimize both Sphere functions. Thus, when performing multi-objective optimization, the outcome comprises a series of Pareto-optimal values but not a single optimal value, which simulates the trade-offs in real-world design challenges. The range of the parameters is $[0, 1]$ and the range of the objective values is roughly $[-1, 1]$.

Finally, when obtaining the output (y) from the Sphere functions, we introduce a noise value for mimicking humans' noisy performance. The noise value is sampled from a Gaussian distribution, denoted as $Noise \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and $\sigma = 0.05$.

B.2 Generating synthetic users and user groups

In our simulations, we shift and scale the *base function* to create a group of testing functions, which is a common approach [24][12]. We also call these generated testing functions as *user functions* or *synthetic users*. Each created user function can be seen as a unique user as it has its own set of Pareto-optimal parameter settings and the corresponding objective values. More specifically, we shift every parameter value in the *base function* (x_1, x_2, x_3, x_4) by different amounts. Figure 9(a) shows as an example. The magnitudes of these shifts are sampled from a uniform distribution within given ranges. We can formally denote this shifting as $x'_n = x_n + \delta_n$, where x_n is the original parameter value, x'_n is the shifted value, $n \in \{1, 4\}$ (each represents a parameter), and the shifting for each parameter is $\delta_n \sim U(-\frac{shift\_range}{2}, \frac{shift\_range}{2})$.

In addition to shifting the *base function*, we also scale the objective function values to further generate diversity. A scalar is directly multiplied by the objective value; see Figure 9(b) as an example. Such a scalar is drawn from a uniform distribution where 1 is the center. We denote this function scalar as $\mathcal{S} \sim U(1 - \frac{scale\_range}{2}, 1 + \frac{scale\_range}{2})$. When the scalar (S) exceeds 1, the maximum value of the function will go beyond 1. Conversely, if S is less than 1, it reduces the function output value. An example is illustrated in Figure 9 (b). This scaling simulates the different levels of performance exhibited by different users.

To simulate different scenarios where the user groups have various levels of similarities, we sample function shifting (δ) and scalar $(\mathcal{S})$ from different ranges ($shift\_range$ and $scale\_range$). In the below simulations, we deployed different $shift\_range$ and $scale\_range$. A small range naturally generates a group of functions with the highest similarities, simulating a highly similar user group. An illustrated example is illustrated in Figure 9(c). Conversely, a

[11] See https://www.sfu.ca/~ssurjano/spheref.html.

[12] The first figure in https://botorch.org/tutorials/meta_learning_with_rgpe also presents an example of shifting and scaling.

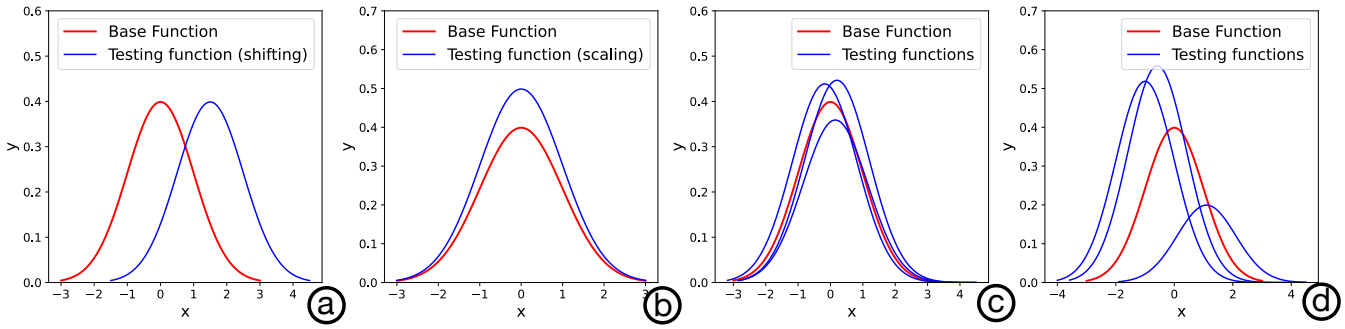

Figure 9: Here we use a typical Gaussian distribution to illustrate how we generate a group of *testing functions* by shifting and scaling a *base function*: (a) demonstration of shifting a *base function*, (b) demonstration of scaling a *base function*, (c) an example user group with a higher similarity, (d) an example user group with higher diversity.

large range leads to a group of highly diverse functions, mimicking a group of very diversified users. See Figure 9(d) for an illustrated example.

In each of the following simulations, we randomly sample 20 shifting (δ) and scaling ($\mathcal{S}$) to create 20 distinct synthetic users, each representing a user. Among these 20 synthetic users, 10 of them are designated as "population users." We perform population modeling on these users. The remaining 10 functions are considered "new users." TAF$^+$, is subsequently applied to these new synthetic users.

B.3 Simulation 1: Validating the robustness of TAF$^+$ under various OBJECTIVE WEIGHT settings

One important feature of our TAF$^+$ is that it allows the designers or developers to dynamically assign the OBJECTIVE WEIGHTS when performing weighted-sum optimization in the adaptation phase. Simulation 1 aims to investigate whether TAF$^+$ can maintain stable performance across varying OBJECTIVE WEIGHTS. Note that the three Sphere functions in our *base function* have distinct optimal parameter values, and two parameters (x_2, x_3) are shared by two Sphere functions. Thus, altering the OBJECTIVE WEIGHTS in weighted-sum optimization results in different sets of optimal parameter settings. Since our focus is on assessing the impact of OBJECTIVE WEIGHT assignments, we intentionally chose the smallest sample range to create the user group: both $shift\_range$ and $scale\_range$ are set to be 0.01.

Step 1: Population modeling via a user data collection. Following our workflow (see subsection 4.2), we performed multi-objective BO to construct 10 population models[13]. The progress is plotted in Figure 10.

Step 2 & 3: OBJECTIVE WEIGHT and decay hyperparameter setting. To test the performance of TAF$^+$ under a wide spectrum of OBJECTIVE WEIGHTS, we deliberately test 6 sets of OBJECTIVE WEIGHTS ($[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$, $[0.33, 0.33, 0.34]$, $[0.5, 0.3, 0.2]$, and $[0.3, 0.5, 0.2]$) when deploying TAF$^+$ on the new users (functions). Since this group of users has high similarity, there is no population model decay applied in this simulation (for details regarding such a decay, please see subsection 4.1).

Adaptation: Evaluating TAF$^+$ on new users. We applied TAF$^+$[14] on 10 new users (unseen user functions). We further applied standard BO[15] and random sampling as baseline conditions. The resulting performances of TAF$^+$ when OBJECTIVE WEIGHTS are set

[13]The hyperparameter setting of multi-objective BO is the same as introduced in subsection 6.1 with 20 initial random samples followed by 20 optimization iterations.
[14]The hyperparameter settings are similar to subsection 7.2 with 256 random sampling when calculating TAF$^+$ values.
[15]The hyperparameter setting is as introduced in subsection 7.2 of the main paper with 20 initial random sampling and 20 optimization iterations.

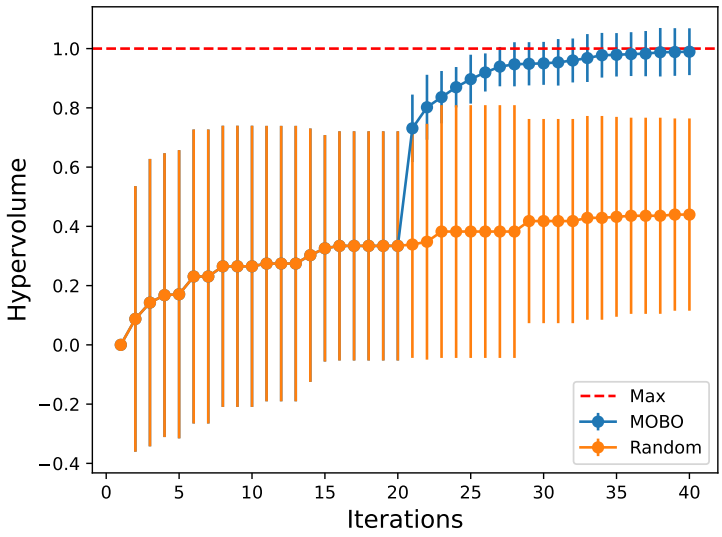

Figure 10: The hypervolume per iteration during the population modeling in simulation 1. The random condition was to serve as a baseline. The error bar denotes a 95% confidence interval. The global maximum hypervolume was determined by a grid search when there was no noise involved.

as $[1, 0, 0]$, $[0, 1, 0]$ (Figure 11), $[0, 0, 1]$, $[0.33, 0.33, 0.34]$ (Figure 12), $[0.5, 0.3, 0.2]$, and $[0.3, 0.5, 0.2]$ (Figure 13) are plotted below.

Findings. Overall, TAF$^+$ performs stably across different OBJECTIVE WEIGHT configurations. It consistently converges to the global maximum value with 3 or 4 iterations. This result highlights the advantage of using multi-objective BO in population modeling: it effectively allows the designers to tune the OBJECTIVE WEIGHTS when needed while maintaining high efficiency when deploying TAF$^+$. Conversely, such flexibility would not exist if the population modeling phase deployed weighted-sum single-objective BO.

B.4 Simulation 2: Validating the effectiveness of TAF$^+$ under different user group similarities

In the second simulation, we aim to understand the performance of TAF$^+$ when dealing with different levels of similarities in the user group. To that end, we modify the sampling range of shifting ($shift\_range$) and scaling ($scale\_range$) when generating the testing functions. Different sampling ranges will result in functions of different levels of similarities. The sampling ranges in this simulation are 0.05, 0.1, 0.2, and 0.3. Particularly, a shifting range of 0.3 can create a huge diversity, given that the whole parameter range is only 1. Similar to Simulation 1, we sampled 20 functions for each sampling range; 10 serve as population users, and the remaining 10 serve as new users.

Step 1: Population modeling via user data collection. Similar to the previous simulation, for each sampling range, we performed multi-objective BO to construct 10 population models[16]. The resulting hypervolumes are plotted in Figure 14 and Figure 15. Despite the increasingly higher variations between functions (indicated by the error bar), multi-objective BO can effectively explore the Pareto frontier for each group.

Step 2 & 3: OBJECTIVE WEIGHT and decay hyperparameter setting. To focus on investigating the effect of user similarity, we fixed the OBJECTIVE WEIGHT at $[0.3, 0.5, 0.2]$ throughout this simulation. We followed the workflow (see subsection 6.3) to optimize the decay hyperparameter settings (d_1 and d_2) For sampling range 0.05, the optimal hyperparameter setting is $(d_1, d_2) = (7, 0.1)$. For sampling range 0.1, the optimal hyperparameter setting is

[16]The hyperparameter setting of multi-objective BO is the same as introduced in subsection 6.1 with 20 initial random samples followed by 20 optimization iterations.

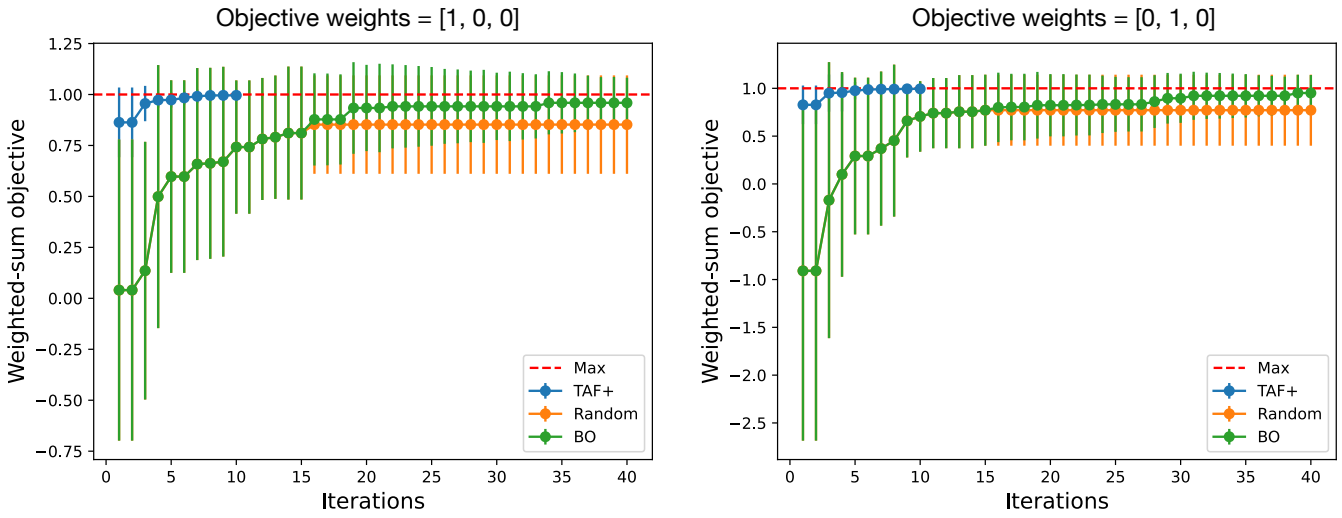

Figure 11: The resulting performance of TAF$^+$ when the OBJECTIVE WEIGHTS are $[1, 0, 0]$ and $[0, 1, 0]$, and the population modeling has δ and $\mathcal{S}$ are sampled from range of 0.01. The error bar indicates a 95% confidence interval. The global maximum objective value was determined by a grid search when there was no noise involved.

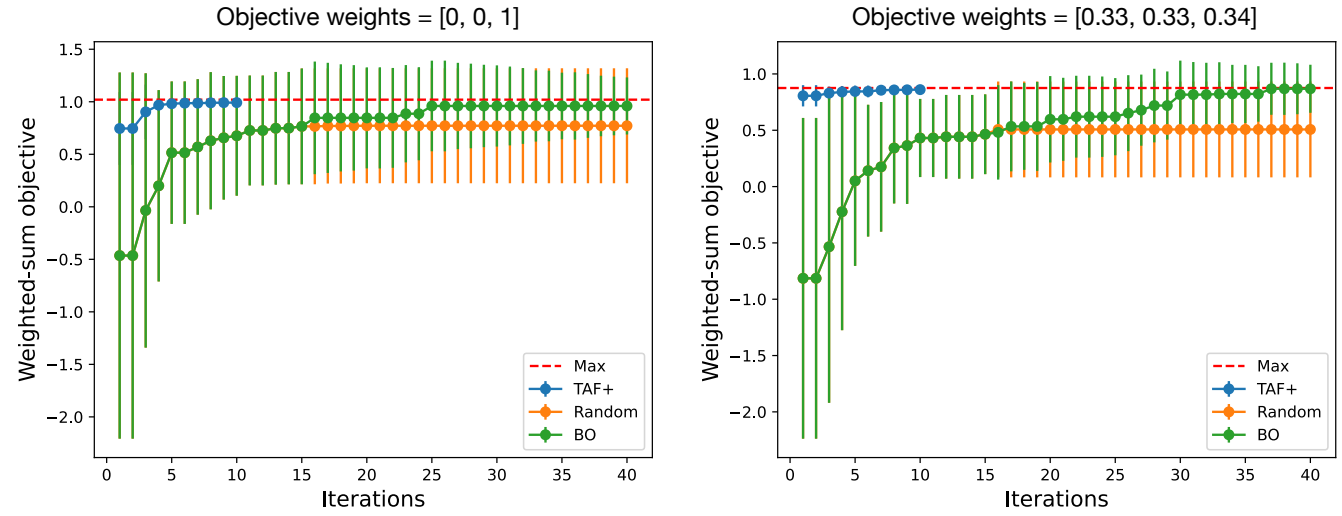

Figure 12: The resulting performance of TAF$^+$ when the OBJECTIVE WEIGHTS are $[0, 0, 1]$ and $[0.33, 0.33, 0.34]$, and the population modeling has δ and $\mathcal{S}$ are sampled from range of 0.01. The error bar indicates a 95% confidence interval.

$(d_1, d_2) = (4, 0.1)$. For sampling range 0.2, the optimal hyperparameter setting is $(d_1, d_2) = (4, 0.2)$. For sampling range 0.3, the optimal hyperparameter setting is $(d_1, d_2) = (3, 0.2)$. These optimal decay settings show that as the user diversity increases, the decay should happen earlier and faster to allow for better adaptation of the new user's characteristics.

Adaptation: Evaluating TAF$^+$ on new users. We applied TAF$^+$ on 10 new user functions. Standard BO and random sampling are set as baselines. The resulting performances of TAF$^+$ are presented in Figure 16 and Figure 17.

Findings. TAF$^+$ overall performs well across different sampling ranges. Upon closer examination, we noted that greater diversity, such as sampling from ranges of 0.2 or 0.3, results in a lower starting performance for TAF$^+$. This outcome is natural, as higher sampling ranges introduce potentially larger differences between the population users and the new users, necessitating more iterations for TAF$^+$ to adapt to the new user functions. However, despite the lower starting point, TAF$^+$ still converges to the maximum objective value within 10 iterations. This highlights TAF$^+$'s potential even when being applied to an interaction where users exhibit high diversity in their preferences and performances.

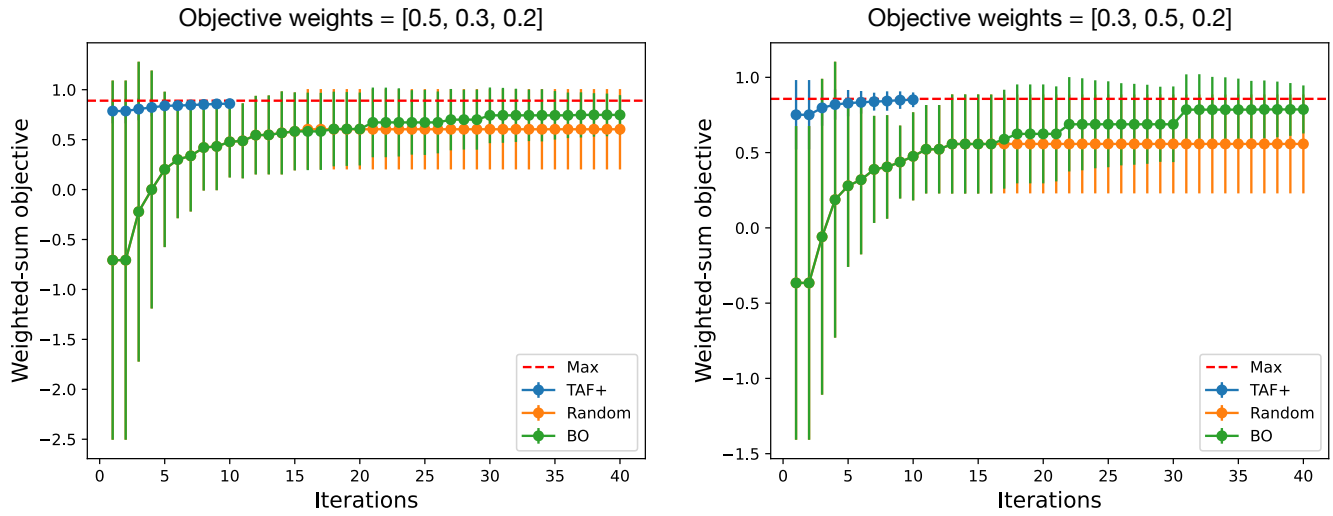

Figure 13: The resulting performance of TAF$^+$ when the OBJECTIVE WEIGHTS are $[0.5, 0.3, 0.2]$ and $[0.3, 0.5, 0.2]$, and the population modeling has δ and $\mathcal{S}$ are sampled from range of 0.01. The error bar indicates a 95% confidence interval.

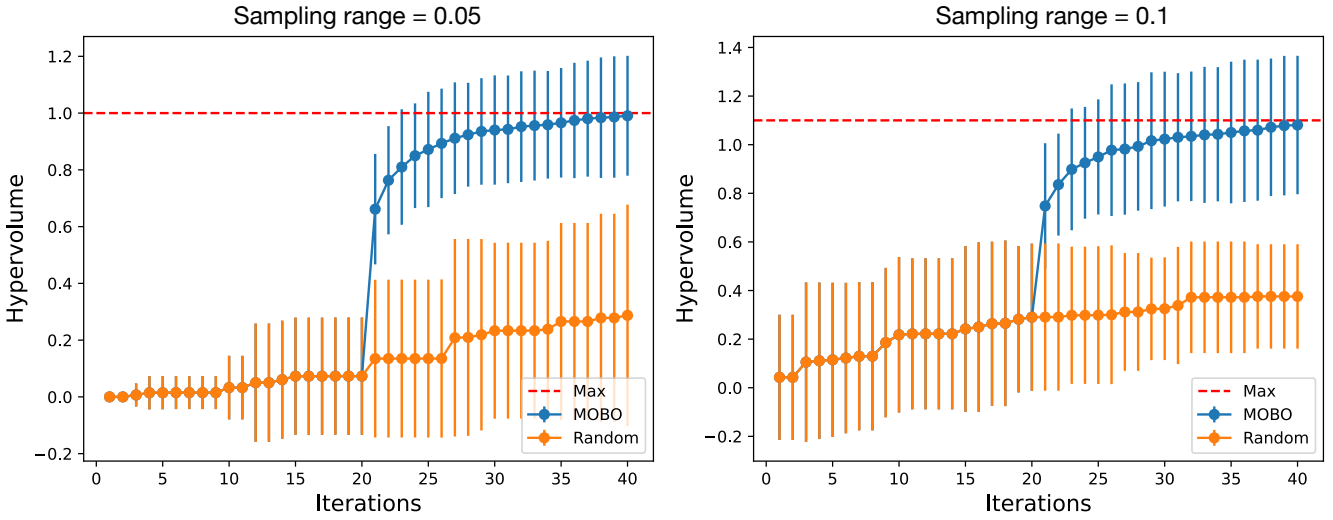

Figure 14: The hypervolume per iteration during the population modeling in Simulation 2. The random condition was to serve as a baseline. The error bar denotes a 95% confidence interval, and the maximum volume is determined by a thorough grid search.

B.5 Simulation 3: Validating the generalizability of TAF$^+$ with different testing functions

From the previous two simulations, we demonstrated the consistent performance of TAF$^+$ when facing the three-sphere *base function*. In the third simulation, we investigate the efficacy of TAF$^+$ when facing other, potentially more challenging, *base functions*. These different base functions are introduced, simulating different interactions. There are three *base functions* in this simulation:

- *Relocated Spheres*: This *base function* is similar to the one used in the previous simulations, which contains 3 Sphere functions. However, the Spheres here have different and further apart center locations. See Table 7.
- *Spheres + Branin*: This *base function* has two Sphere functions and a Branin[17]. Branin is a commonly used function for testing optimization and it is considered more challenging than Sphere. Details are in Table 8.

[17] Please see http://www.sfu.ca/~ssurjano/branin.html for more details. We normalized the parameter space into $[0, 1]$ range, and reversed the function so that it becomes a maximization problem.

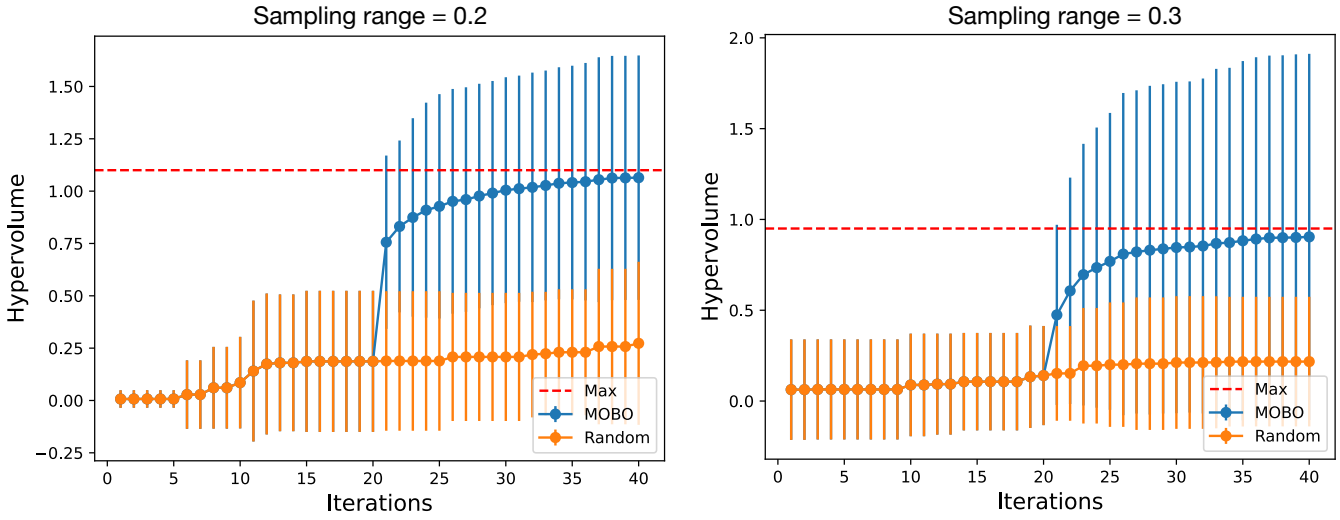

Figure 15: The hypervolume per iteration during the population modeling in Simulation 2. The random condition was to serve as a baseline. The error bar denotes a 95% confidence interval, and the maximum volume is determined by a thorough grid search.

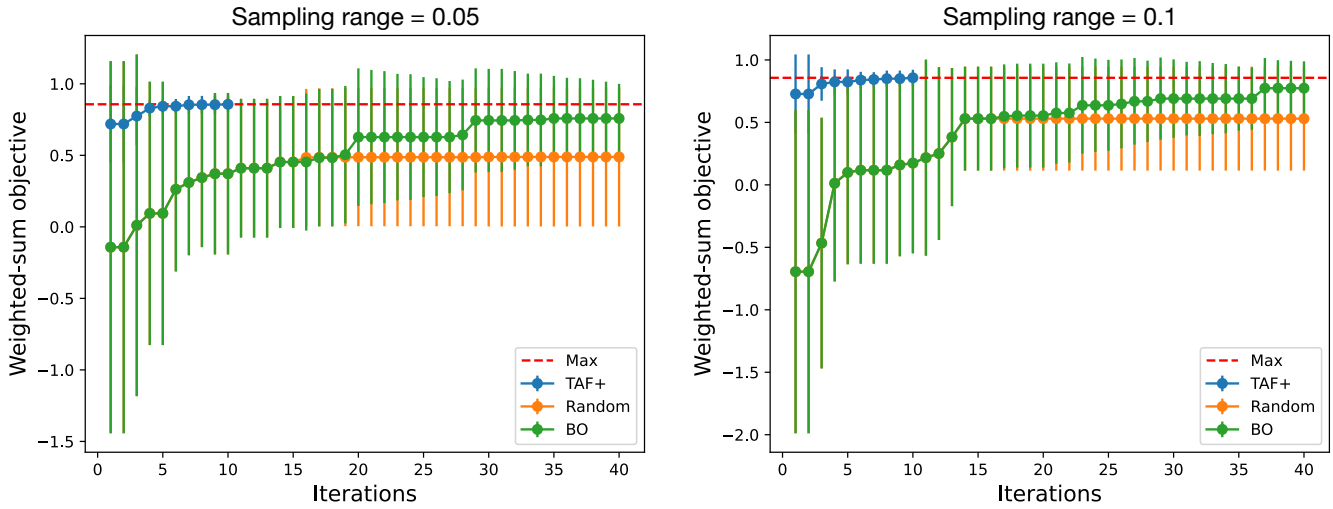

Figure 16: The resulting performance of TAF$^+$ when the functions' shifting (δ) and scaling (S) are sampled from range of 0.05 and 0.1. The error bar indicates a 95% confidence interval.

- *Sphere + Branin + Hartmann 3D*: This *base function* is further complicated. It contains a Sphere, a Branin, and a Hartmann 3D[18]. Hartmann 3D is a function that takes 3 parameters as input, and it is also a widely recognized testing function. Note that X_2 in this function is shared by three functions, which added more complexity to this task. Details of this function are in Table 9.

[18]Please see https://www.sfu.ca/~ssurjano/hart3.html for more details. We reversed the function so that it becomes a maximization problem.

We set the sampling range as 0.1 for both function shifting (δ) and scaling (S) when generating user functions. Same as in previous simulations, we sampled 20 user functions; 10 serve as population users, and the remaining 10 serve as new users.

Step 1: Population modeling via user data collection. We followed the same procedure, using multi-objective BO, to construct population models. The resulting hypervolume is plotted in Figure 18. Because *Spheres + Branin* and *Sphere + Branin + Hartmann 3D* functions are more challenging than the three-Sphere function,

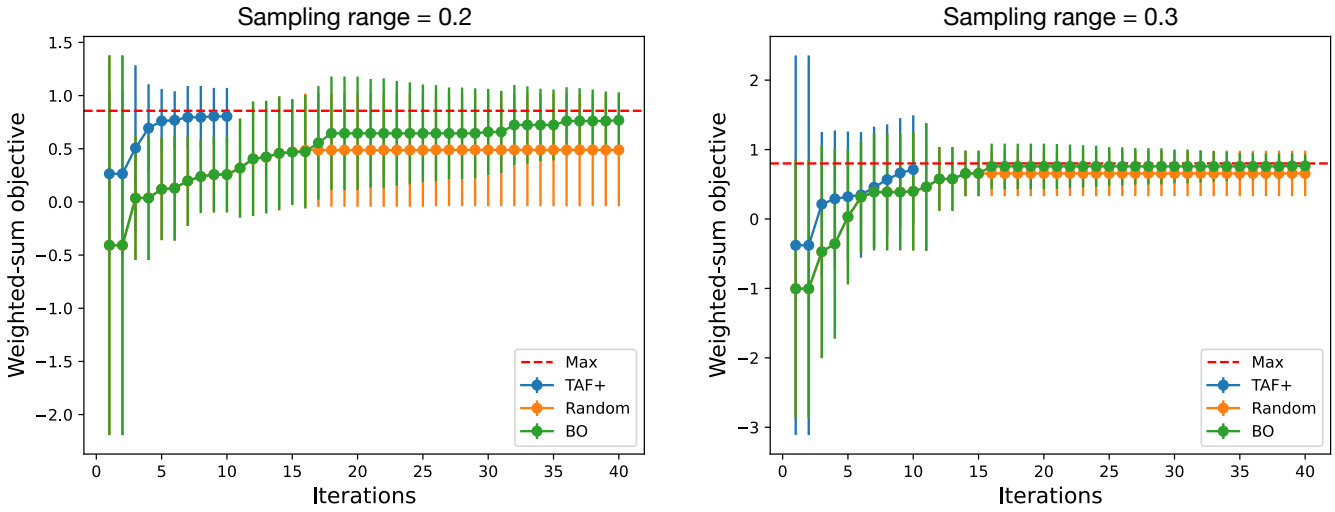

Figure 17: The resulting performance of TAF$^+$ when the functions' shifting (δ) and scaling ($\mathcal{S}$) are sampled from range of 0.2 and 0.3. The error bar indicates a 95% confidence interval.

Parameters	Corresponding Objective Functions	Range
x_1	Sphere 1, center = (0.55, 0.3)	[0, 1]
x_2	Sphere 1, center = (0.55, 0.3) & Sphere 2 (center = (0.8, 0.35)	[0, 1]
x_3	Sphere 2, center = (0.8, 0.35) & Sphere 3, center = (0.7, 0.4)	[0, 1]
x_4	Sphere 3, center = (0.7, 0.4)	[0, 1]

Table 7: The parameters of the Relocated Spheres base function.

Parameters	Corresponding Objective Functions	Range
x_1	Sphere 1, center = (0.55, 0.3)	[0, 1]
x_2	Sphere 1, center = (0.55, 0.3) & Branin	[0, 1]
x_3	Branin & Sphere 2, center = (0.7, 0.4)	[0, 1]
x_4	Sphere 2, center = (0.7, 0.4)	[0, 1]

Table 8: The parameters of the Spheres + Branin base function.

Parameters	Corresponding Objective Functions	Range
x_1	Sphere 1, center = (0.55, 0.3)	[0, 1]
x_2	Sphere 1, center = (0.55, 0.3) & Branin & Hartmann 3D	[0, 1]
x_3	Branin & Hartmann 3D	[0, 1]
x_4	Hartmann 3D	[0, 1]

Table 9: The parameters of the Sphere + Branin + Hartmann 3D base function.

we extended the optimization iterations to ensure the quality of the population models.

Step 2 & 3: Objective weight and decay hyperparameter setting. Similar to Simulation 2, we set the objective weight at $[0.3, 0.5, 0.2]$ in this simulation. We followed the workflow to optimize the decay hyperparameter settings, and the resulting optimal setting is $(d_1, d_2) = (4, 0.1)$.

Adaptation: Evaluating TAF$^+$ on new users. We deployed TAF$^+$ on 10 new users. The resulting performances of TAF$^+$ compared against BO and random sampling are presented in Figure 19.

Findings. The results highlight the promising performance of TAF$^+$ when facing different sets of functions, demonstrating the potential generalizability of TAF$^+$ for different scenarios and interactions. It is also worth noting that BO encounters challenges in efficiently reaching optimal objective values when applied to more complex functions, such as mixing Branin, Sphere, and Hartmann 3D. However, with the aid of population models, TAF$^+$ is able to leverage the prior experience to address such complex functions efficiently; which further showcases the generalizability of our approach when facing challenging optimization problems.

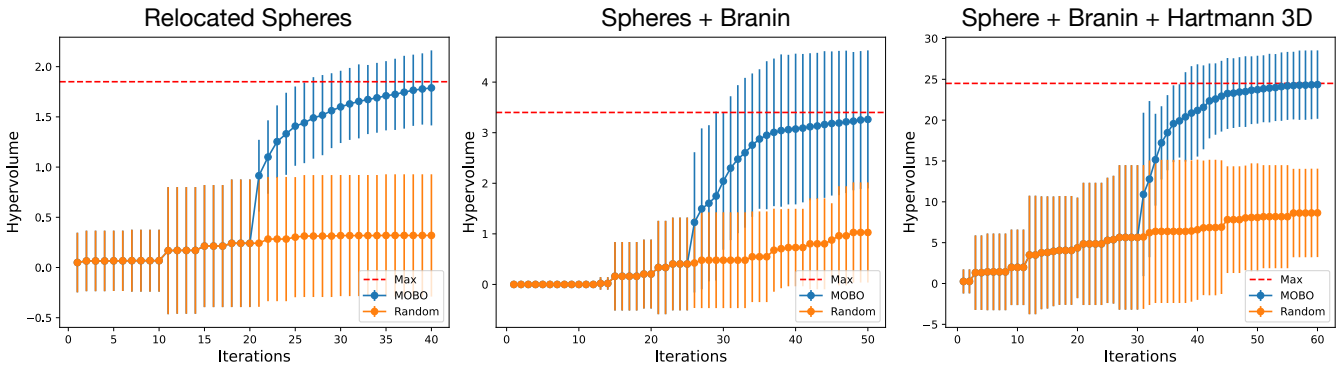

Figure 18: The resulting performance of TAF$^+$ when the functions' shifting (δ) and scaling ($\mathcal{S}$) are sampled from range of 0.05 and 0.1. The error bar indicates a 95% confidence interval.

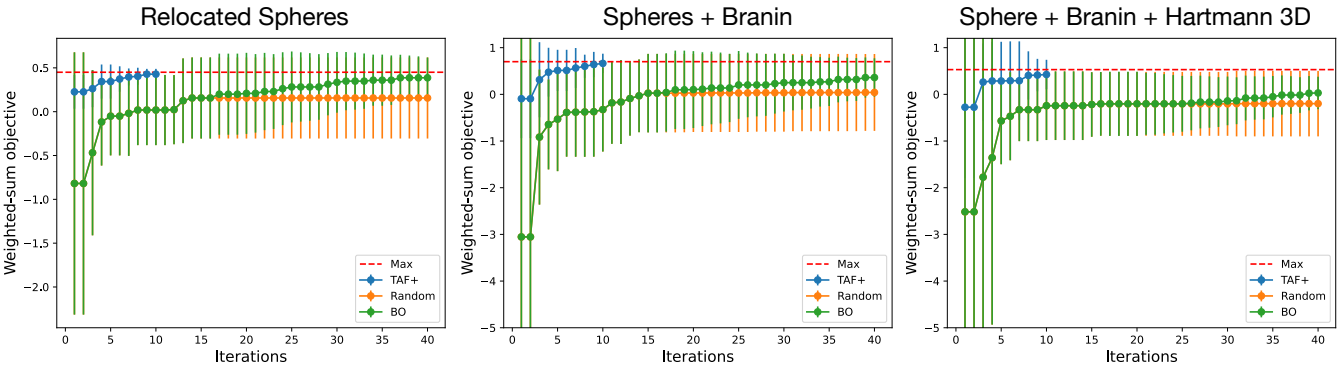

Figure 19: The resulting performance of TAF$^+$ when the functions' shifting (δ) and scaling ($\mathcal{S}$) are sampled from ranges of 0.2 and 0.3. The error bar indicates a 95% confidence interval.

B.6 Simulation 4: Investigating the performance of TAF$^+$ under different population model sizes

In this simulation, we aim to evaluate the performance of TAF$^+$ when it has various numbers of population models. We utilized the 3-Sphere *base function*, the same as Simulations 1 and 2 (see Table 6). The sampling ranges (*shift_range* and *scale_range*) are set to be 0.2, and the results are presented in Figure 20. We do not present the details of population modeling and decay hyperparameter setting as they are the same as in previous simulations 1 and 2.

Adaptation: Evaluating TAF$^+$ with different numbers of population models. The resulting performance over iterations are shown in Figure 20. Overall, with a higher number of the population model, TAF$^+$ potentially has a better starting performance. However, regardless of the number of population models, with sufficient iterations and proper decay hyperparameter configuration, TAF$^+$ can still converge toward optimality.

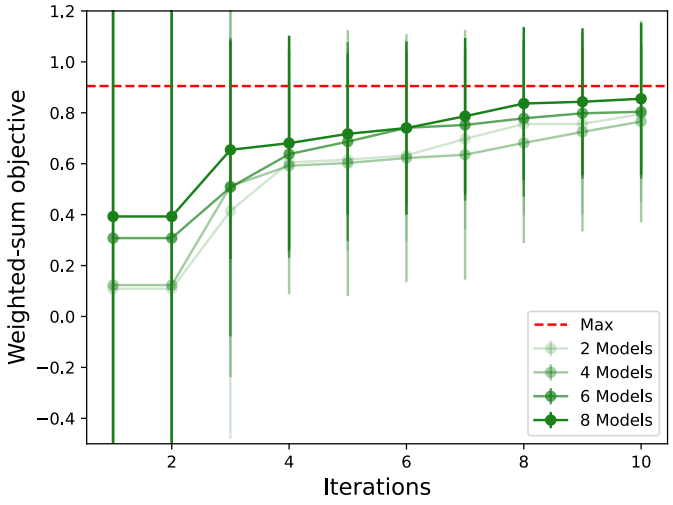

Figure 20: The resulting performance of TAF$^+$ when the number of population models varies.

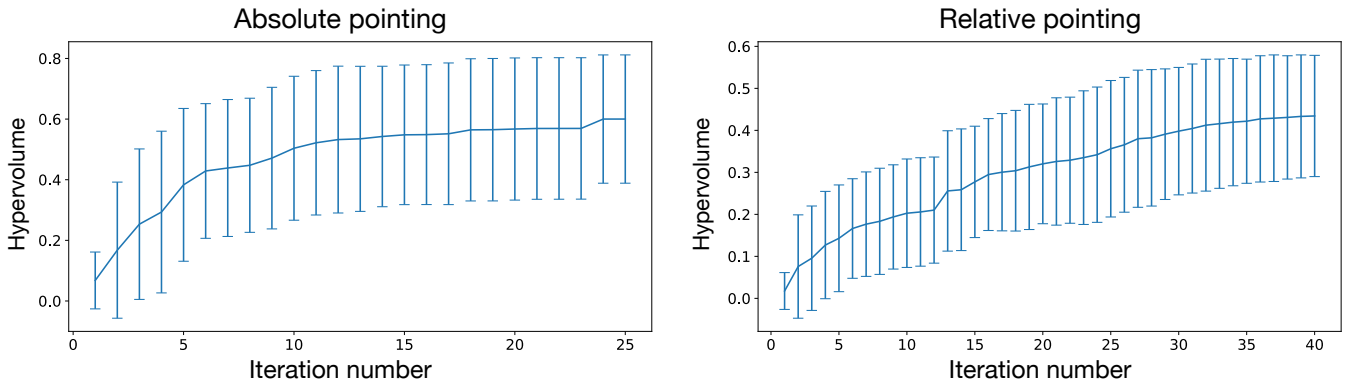

Figure 21: The resulting hypervolume per iteration for both interactions during population modeling. The error bar denotes a 95% confidence interval.

B.7 Overall findings of the simulations

The above simulations collectively demonstrate the robustness of TAF$^+$ under diverse conditions, highlighting its adaptability and generalizability.

Simulation 1 examines TAF$^+$ under varying OBJECTIVE WEIGHTS, and the positive results highlight the flexibility of adjusting the OBJECTIVE WEIGHTS when deploying TAF$^+$. This flexibility is particularly useful when conducting weighted-sum optimization for real-world interactions, where designers may need to fine-tune the weighting on objectives. Simulation 2 assesses TAF$^+$'s performance under different levels of user diversity, and the results showed TAF$^+$'s effectiveness under diversified user groups, highlighting TAF$^+$'s ability to adapt to different user performances and preferences. Simulation 3 validates TAF$^+$'s generalizability across a variety of base functions. Despite the increasing challenges of the functions, TAF$^+$ retains promising performance whereas standard BO may not be able to converge efficiently. Simulation 4 shows the impact of different numbers of population models. While the less population models may lead to a lower initial performance, TAF$^+$ can converge more efficiently than standard BO. To summarize, the results and findings provide evidence that TAF$^+$ is adaptable and capable of handling a wide spectrum of problems with different levels of difficulties and conditions.

Finally, to further assess the computation complexity, the first 3 simulations with 10 population models were run on a ThinkPad X1 Carbon, with Ubuntu 22.04.3 OS and an Intel i7-10150U CPU. Throughout these simulations, TAF$^+$'s computation time for each iteration is 6.17 seconds ($s.d.$ = 1.17).

C HYPERVOLUME INCREASE IN POPULATION MODELING

Figure 21 shows the hypervolume of both interactions at each iteration during the population modeling.

D EXTENDED SIMULATION FOR THE DECAY HYPERPARAMETER OPTIMIZATION

To validate the convergence of the optimization methods, we extended the decay hyperparameter optimization. Figure 22 shows the extended simulation for the decay hyperparameter optimization. The result shows that TAF+, TAF, and BO all converge near the global max given enough iterations.

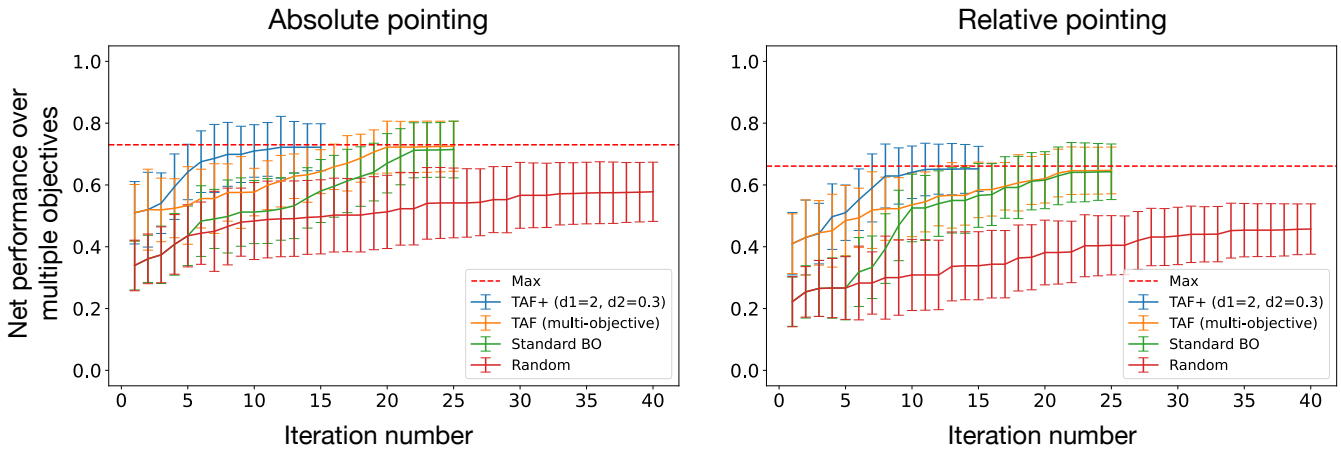

Figure 22: The resulting performance under simulation. The max performance was derived by a grid search.

E DETAILS OF THE EVALUATION

Here, we provide the detailed numbers of the evaluation, which complements Figure 8. Table 10 and Table 11 list all the mean and standard deviation of weighted-sum performance in absolute pointing and relative pointing, respectively. We also found the overall performance is better than the estimated performance in step 3 (decay hyperparameter optimization, and see Figure 7). We examined the performance we gathered in step 1 (population modeling) and compared it to the performance gathered in adaptation[19]. With t-test, we found a significant difference ($t(23) = -3.576$, $p = 0.02$) between the population user group (mean = 72.87, $s.d. = 0.114$) and the adaptation user group (mean = 0.87, $s.d. = 0.11$). The difference in the performance levels may cause the overall difference in absolute pointing between Figure 7 and Figure 8. The positive result in Figure 8 also validates the efficacy of TAF$^+$ when the users have different levels of performance.

Table 10: The mean and standard deviation of the weighted-sum performance at each procedure in absolute pointing.

# of Iteration	Meta-BO	BO	Manual
1	0.64 (s.d. = 0.09)	0.31 (s.d. = 0.08)	0.42 (s.d. = 0.1)
2	0.71 (0.08)	0.56 (0.07)	0.57 (0.08)
3	0.79 (0.05)	0.62 (0.06)	0.70 (0.07)
4	0.81 (0.03)	0.70 (0.05)	0.61 (0.07)
5	0.83 (0.03)	0.71 (0.05)	0.66 (0.07)
6	0.85 (0.03)	0.71 (0.05)	0.72 (0.06)
7	0.86 (0.03)	0.72 (0.05)	0.74 (0.06)
8	0.87 (0.02)	0.75 (0.05)	0.75 (0.05)
9	0.87 (0.02)	0.75 (0.05)	0.78 (0.05)
10	0.87 (0.02)	0.75 (0.05)	0.80 (0.04)

Table 11: The mean and standard deviation of the weighted-sum performance at each procedure in relative pointing.

# of Iteration	Meta-BO	BO	Manual
1	0.27 (s.d. = 0.07)	0.14 (s.d. = 0.05)	0.31 (s.d. = 0.05)
2	0.46 (0.05)	0.35 (0.07)	0.42 (0.06)
3	0.60 (0.03)	0.37 (0.06)	0.50 (0.05)
4	0.62 (0.04)	0.43 (0.05)	0.50 (0.05)
5	0.66 (0.03)	0.53 (0.04)	0.50 (0.05)
6	0.68 (0.02)	0.55 (0.04)	0.58 (0.03)
7	0.68 (0.02)	0.59 (0.04)	0.59 (0.03)
8	0.68 (0.02)	0.61 (0.03)	0.61 (0.03)
9	0.68 (0.02)	0.61 (0.03)	0.63 (0.03)
10	0.70 (0.02)	0.61 (0.03)	0.64 (0.04)

[19]For each user, we first derived the highest weighted-sum performance, and then we compared the optimal performance of the population users and the users in the adaptation phase.

Table 12: The mean and standard deviation of the raw NASA-TLX scores in absolute pointing. One-way repeated-measures ANOVAs showed that the only significant difference was found in the sixth question where both Meta-BO and BO led to significantly lower scores than the Manual procedure.

Questions	Meta-BO	BO	Manual
1. How mentally demanding was the task?	6.55 (s.d. = 4.76)	5.64 (s.d. = 3.67)	8.64 (s.d. = 5.46)
2. How physically demanding was the task?	7.91 (4.93)	6.45 (2.73)	9.18 (4.83)
3. How hurried or rushed was the pace of the task?	5.18 (4.69)	3.73 (3.32)	4.91 (3.67)
4. How successful were you in accomplishing what you were asked to do?	4.45 (3.46)	2.91 (1.97)	5.18 (4.14)
5. How hard did you have to work to accomplish your level of performance?	9.09 (4.91)	7.73 (3.04)	9 (5.98)
6. How insecure, discouraged, irritated, stressed, and annoyed were you?*	**3.82 (2.67)**	**3.82 (3.16)**	**8 (5.67)**

F ANALYSIS ON THE PERCEIVED WORKLOAD AND USER EXPERIENCE

We present the detailed result of the NATA-TLX questionnaire in Table 12 and Table 13 for absolute pointing and relative pointing, respectively. We performed one-way repeated measures ANOVA with Greenhouse-Geisser correction on each question individually. The only statistically significant difference was found in the 6th question, which is related to the perceived frustration ("How insecure, discouraged, irritated, stressed, and annoyed were you?"), in absolute pointing ($f(1.26) = 6.735$, $p < 0.05$). Pairwise comparison with Bonferroni correction showed that both Meta-BO and BO led to significantly lower frustration levels than Manual (both $p < 0.05$). In the interview, most participants reflected on the fact that the manual calibration is time-consuming and not bringing a better experience is frustrating and unwanted. We did not find significant differences for the rest questions in both interactions. Overall, these results indicate that Meta-BO delivers a comparable or better experience to the users than other procedures.

F.1 User experience

Based on the user interviews, we learned that most users felt both TAF$^+$ and BO gradually proposed improving designs. Interestingly, six users felt BO proposed abrupt parameter settings. P1 mentioned, "*[...] this round (BO) has more noticeable changes. It suddenly changed speeds (the transfer function) [...] It was a bit unexpected.*" P5 made a similar remark, "*It feels a bit more unstable than the previous ones (TAF$^+$ and Manual). Overall, it still improves, but sometimes it behaves strangely.*" Such abrupt changes were not mentioned with the TAF$^+$ and Manual procedures.

While the performance differences between the techniques are clear, participants found it somewhat difficult to clearly state which technique was better at adapting better and faster. Our experiment involves selecting targets of different distances and sizes, further adding to the challenge of differentiating the techniques subjectively when the user is focused on the task and not on perceiving the differences. Prior work that analyzed user experience in pointing interactions had similar challenges. For example, Casiez and Roussel [11] investigated the impact of various transfer functions. They summarized that the users could not tell the differences between different transfer functions despite the significant differences in performance. Casiez et al. [14] conducted a thorough investigation on the impact of different types of transfer functions, focusing only on performance metrics. Lee et al. [52] compared various transfer function adaptation techniques and found few significant differences when analyzing NASA-TLX scores.

G PERFORMANCE OF EACH PERFORMANCE METRIC

Figure 23 and Figure 24 show the individual performance metrics throughout 10 iterations for absolute pointing and relative pointing, respectively. We ran one-way repeated-measures ANOVA on each iteration across all metrics and found that the meta-BO procedure led to either comparable or significantly better results than the other baselines.

For **normalized completion time** in *absolute pointing*: Meta-BO outperformed BO ($p < 0.01$) at the third, sixth, and eighth iterations; Meta-BO also outperformed Manual at the fourth iteration. For **aiming error** and **trajectory travel distance** in *absolute pointing*: Meta-BO outperformed BO ($p < 0.01$) at the third iteration, and also outperformed Manual at the fourth iteration.

For **normalized completion time** in *relative pointing*: Meta-BO outperformed BO ($p < 0.01$) at the third, sixth, and seventh iterations; Meta-BO also outperformed Manual at the fourth iteration. For **trajectory travel distance** in *relative pointing*: Meta-BO outperformed both BO and Manual ($p < 0.01$) at the third iteration. These significant differences are also marked in Figure 23 and Figure 24.

Table 13: The mean and standard deviation of the raw NASA-TLX scores in relative pointing. No significant differences were found across all questions.

Questions	Meta-BO	BO	Manual
1. How mentally demanding was the task?	9.09 (s.d. = 5.52)	8.36 (s.d. = 4.65)	8.55 (s.d. = 4.39)
2. How physically demanding was the task?	10.09 (5.89)	8.64 (4.25)	8.73 (3.41)
3. How hurried or rushed was the pace of the task?	6.36 (4.90)	5.09 (3.81)	5.18 (3.22)
4. How successful were you in accomplishing what you were asked to do?	4.64 (4.27)	2.91 (1.97)	5.09 (3.70)
5. How hard did you have to work to accomplish your level of performance?	8.27 (4.00)	8.91 (4.48)	10 (2.76)
6. How insecure, discouraged, irritated, stressed, and annoyed were you?	6.36 (4.41)	5.91 (3.36)	6.73 (4.54)

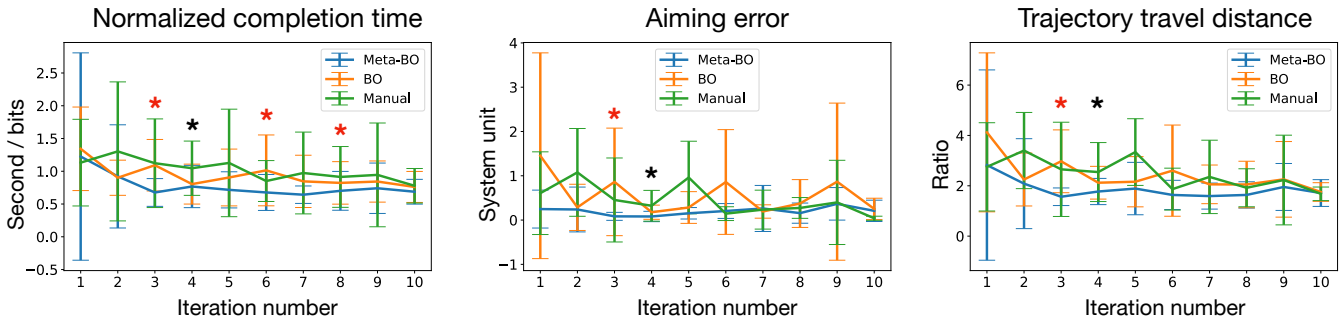

Figure 23: The performance of individual objective functions in absolute pointing. The error bar shows a 95% confidence interval. The red ∗ (meta-BO and BO) and the black ∗ (meta-BO and Manual) signs denote a significant difference between the two procedures at that iteration.

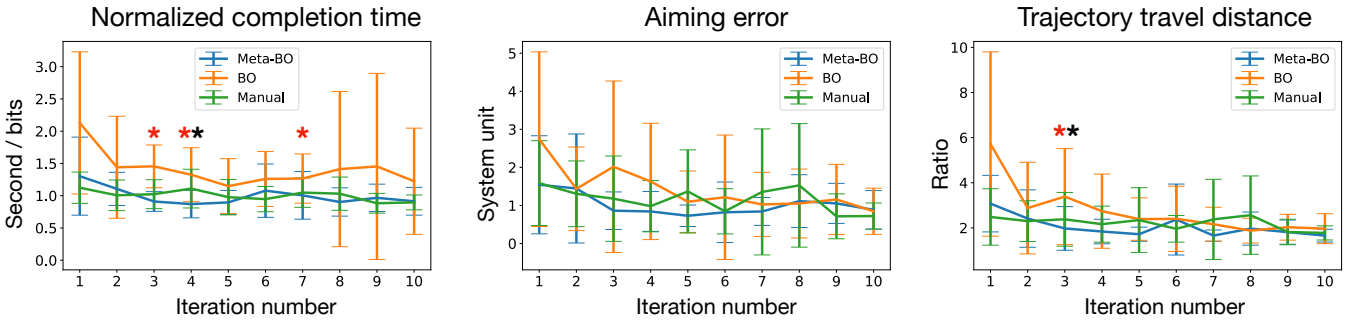

Figure 24: The performance of individual objective functions in relative pointing. The error bar shows a 95% confidence interval. The red ∗ (meta-BO and BO) and the black ∗ (meta-BO and Manual) signs denote a significant difference between the two procedures at that iteration.

H PERFORMANCES AND OPTIMAL PARAMETER SETTINGS OF EACH USER

Figure 25 shows the performances of *absolute pointing* of each user, and Table 14 presents all the users' optimal parameter settings of *absolute pointing*. 10 users start with better performance with meta-BO than with other baselines. Only P5 had a lower initial performance; still, the performance of Meta-BO quickly increased after the first 2 iterations for this user. Upon examining the optimal parameter setting of each user (Table 14), we can see most users' optimal parameter settings have relatively high S_x and S_y, which is aligned with Figure 27. P5 is exceptional since their optimal parameter setting is drastically different from others, which explains their lower initial performance in Figure 25 (also denoted as P5). However, TAF$^+$ can still identify an optimal setting which is different from other participants for P5, highlighting its capacity to deal with drastically different new users.

Figure 26 shows the performances of *relative pointing* of each user, and Table 15 shows the optimal parameter settings for *relative pointing* of each user. 4 users have the highest performance with meta-BO than with other procedures in the first iteration. Later at the 3rd iteration, meta-BO results in the best performances for 8

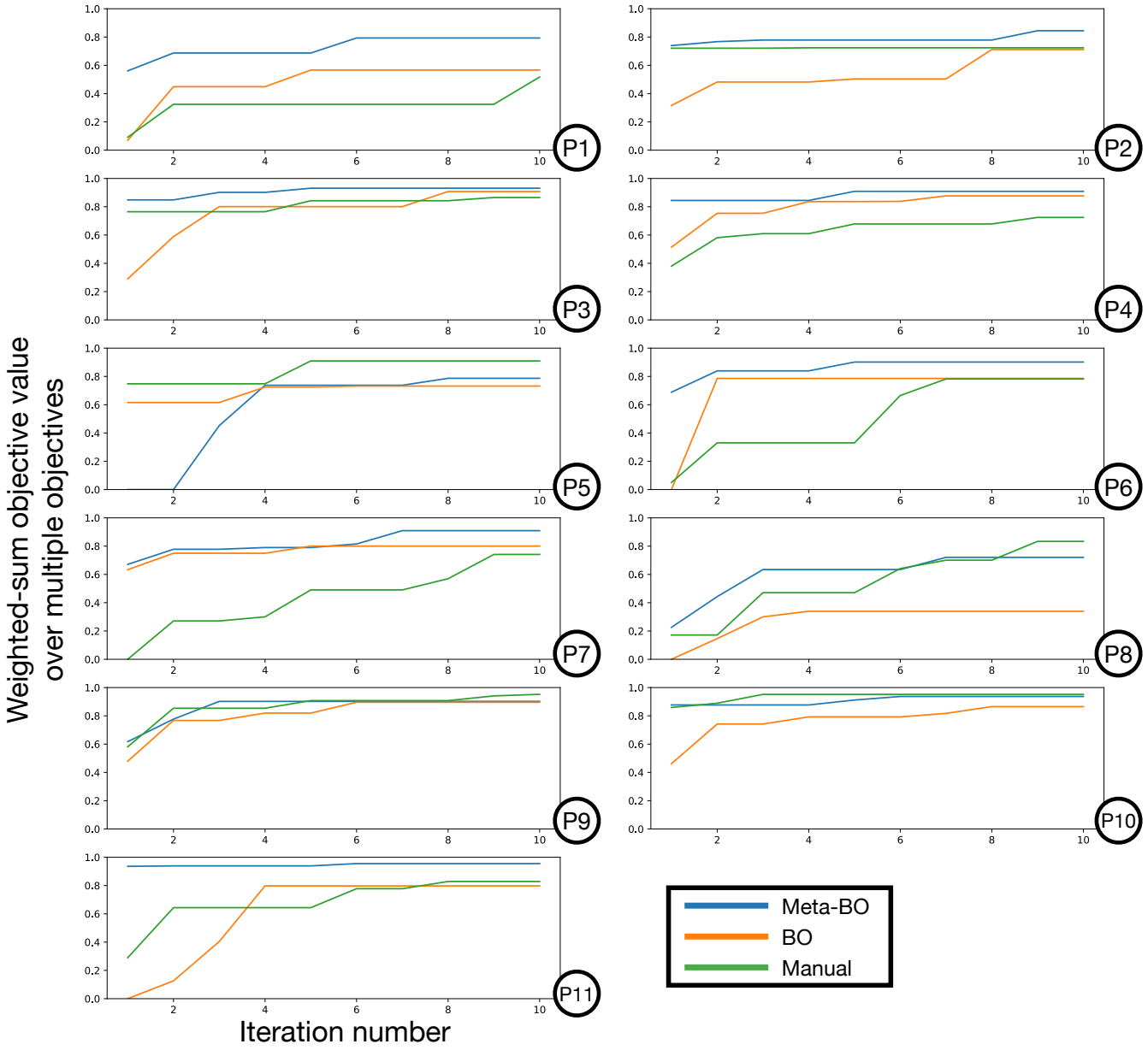

Figure 25: The performances of absolute pointing of each individual user. The x axis is the number of iterations, and the y axis is the weighted-sum final objective value. Note that we have converted the problem into a maximization problem by normalizing the objective functions.

users, which highlights TAF$^+$'S strength in faster adaptation. Table 15 further shows there exist good parameter ranges for most users. For instance, the best range of θ_c and S_1 is around 0.7 to 1. However, P4's optimal setting has a particularly low S_1, which results in a slightly lower starting performance when using TAF$^+$ (P4 in Figure 26). Again, despite the diversity, TAF$^+$ is able to identify the unique optimal setting, showing its ability to handle the diversity within the user group.

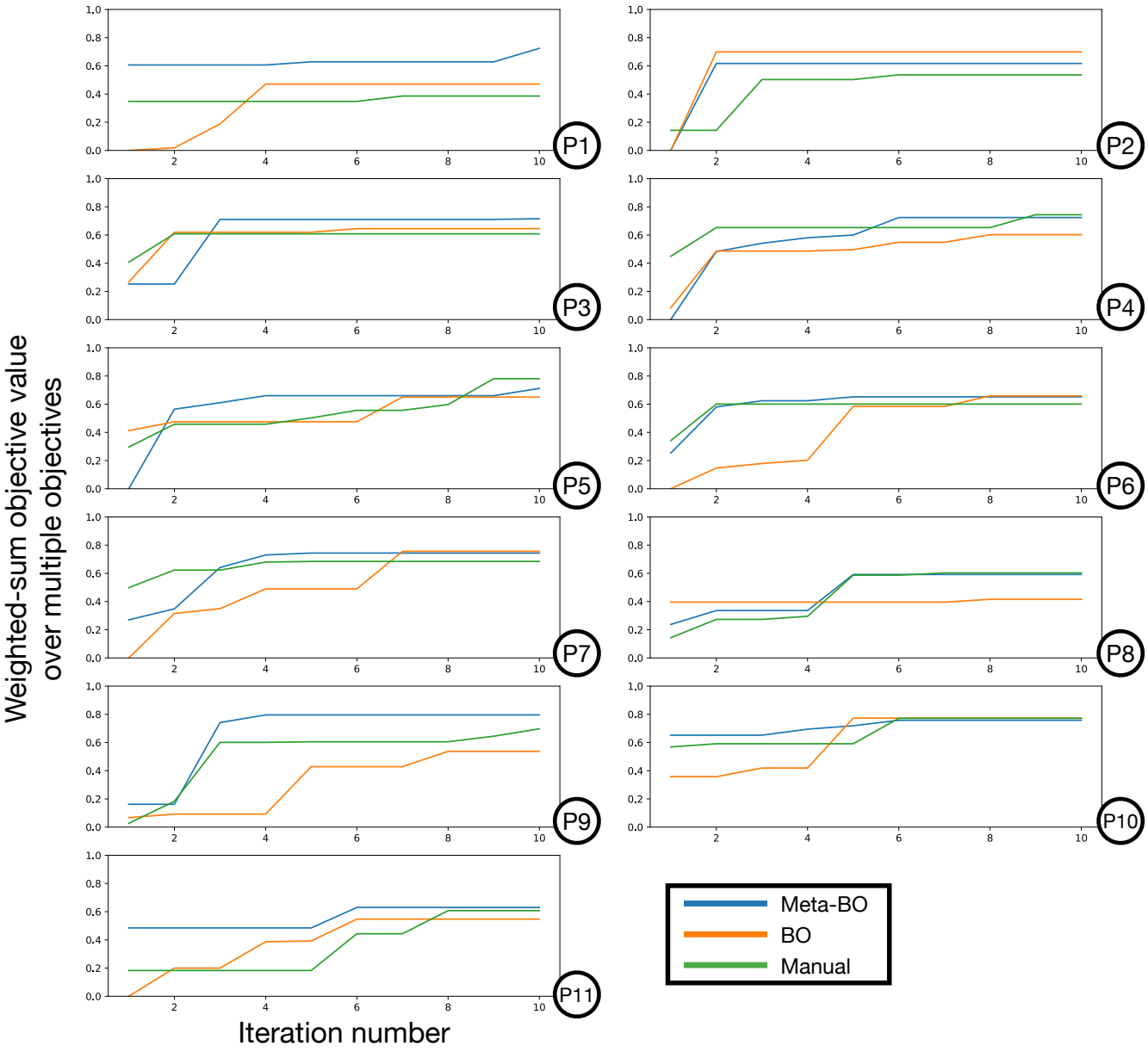

Figure 26: The performances of relative pointing of each individual user. The x axis is the number of iterations, and the y axis is the weighted-sum final objective value. Note that we have converted the problem into a maximization problem by normalizing the objective functions.

User ID	S_x	S_y
P1	0.81	0.86
P2	0.94	1.00
P3	0.78	0.86
P4	0.71	0.78
P5	**0.28**	**0.21**
P6	0.87	0.86
P7	0.86	0.80
P8	0.82	0.94
P9	0.65	0.89
P10	0.67	0.77
P11	0.92	0.94

Table 14: Every user's optimal parameter setting for absolute pointing. Note that we normalized all the parameter values into the $[0, 1]$ range for easier interpretation.

User ID	θ_c	S_1	S_2	S_3
P1	0.58	0.73	0.22	0.49
P2	0.86	0.95	0.47	0.34
P3	0.89	1.00	0.56	0.38
P4	**0.99**	**0.02**	**0.40**	**0.53**
P5	0.69	0.73	0.25	0.40
P6	0.67	0.78	0.26	0.45
P7	0.66	1.00	0.32	0.26
P8	0.73	0.75	0.38	0.55
P9	0.81	0.66	0.37	0.13
P10	0.81	0.82	0.25	0.26
P11	0.79	0.04	0.20	0.62

Table 15: Every user's optimal parameter setting for relative pointing. Note that we normalized all the parameter values into the $[0, 1]$ range for easier interpretation.

I VISUALIZING THE OBJECTIVE FUNCTION OF ABSOLUTE POINTING INTERACTION

To visualize the objective function of absolute pointing, we fitted all the observations (<parameter values, resulting objective value>), across all users and across both TAF$^+$ and BO procedures into a GP. Then, we query the GP's predicted mean. The resulting objective function values are plotted as a heatmap (Figure 27).

We can see that high S_x and S_y values generally allow for better user performance. Also, when S_x and S_y have high differences (indicating unbalanced transfer functions along two axes), the user performance is generally poor.

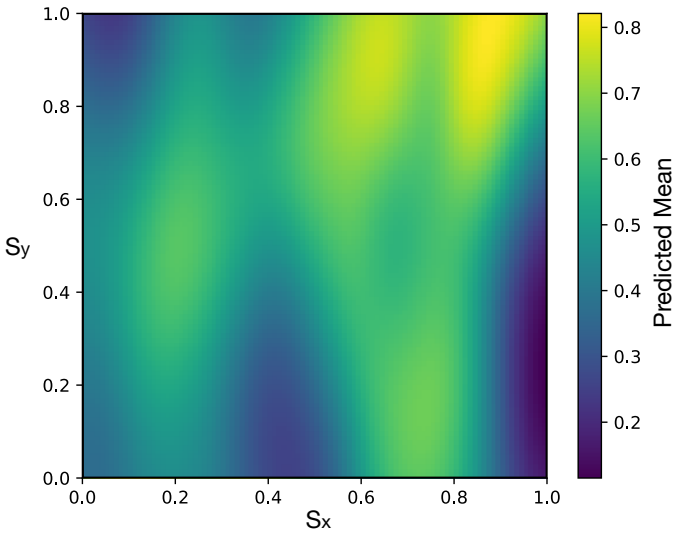

Figure 27: The visualization of the objective function of the absolute pointing.

J SIMULATING MULTI-OBJECTIVE TAF USING EXPECTED HYPERVOLUME INCREASE

Figure 28 shows the hypervolume at each iteration of the simulation. There are three optimization procedures: multi-objective TAF, standard multi-objective BO, and random samples. The multi-objective TAF largely follows our TAF$^+$ setting as described in the paper but changed the base acquisition function from *Expected Improvement* to *Expected Hypervolume Improvement (EHVI)* [19, 22]. With this change, multi-objective TAF identifies the Pareto-optimal parameter settings. In contrast, TAF$^+$ identifies one optimal parameter setting for a weighted-sum objective metric. The baseline multi-objective Bayesian optimization with EHVI was implemented with BoTorch [4][20]. In this simulation, the computation time per iteration ranges from 62 to 95 seconds, conducted on a ThinkPad X1 Carbon running Ubuntu 22.04.3 OS with an Intel i7-10150U CPU. The relatively long computation time highlights the potential challenges of applying multi-objective TAF to fast-paced interactions.

[20]Hyperparameter settings: We used a single-task GP with Matern 5/2 kernel. Similar to Chan et al. [15], we set $q = 1$. We further set 10 optimization restarts for the optimization of the acquisition function, 1024 restart candidates for the acquisition function optimization, and 512 Monte Carlo samples to approximate the acquisition function.

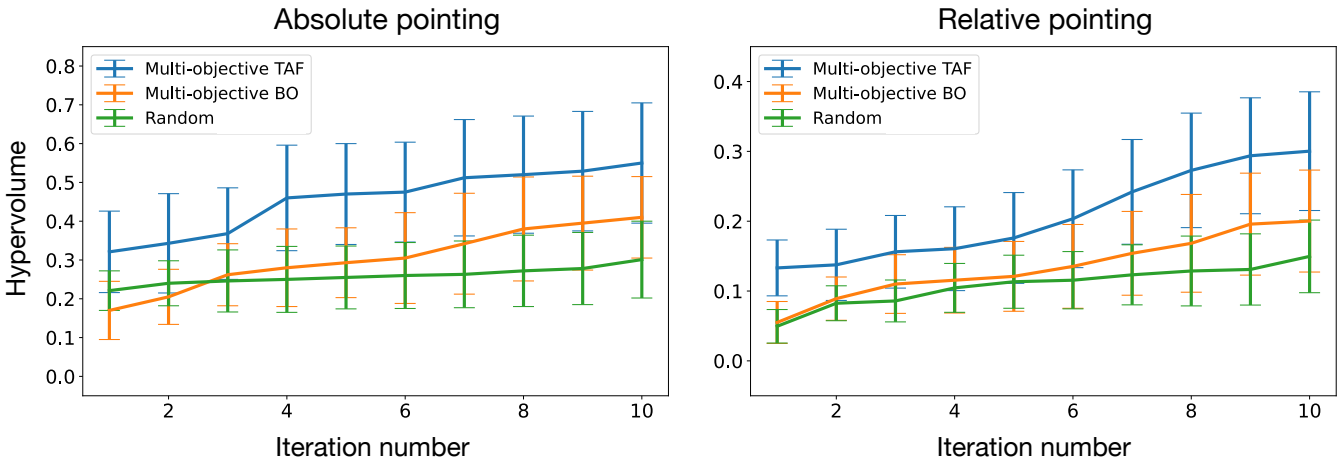

Figure 28: The hypervolume of three different procedures derived via a simulation.