TraceRing: Touchpad-like Pointing with a Single IMU Ring through Personalized Learning

Zhe He
Department of Computer Science and Technology, BNRist, Tsinghua University
Beijing, China
hz23@mails.tsinghua.edu.cn

Weinan Shi*
Department of Computer Science and Technology, Tsinghua University
Beijing, China
swn@tsinghua.edu.cn

Zixuan Wang
Department of Computer Science and Technology, Tsinghua University
Beijing, China
w-zx21@mails.tsinghua.edu.cn

Suya Wu
Department of Computer Science and Technology, Tsinghua University
Beijing, China
wusuya9345@163.com

Xiyuan Shen†
Paul G. Allen School of Computer Science & Engineering | DUB Group, University of Washington
Seattle, USA
xyshen@uw.edu

Chengchi Zhou
Department of Computer Science and Technology, Tsinghua University
Beijing, China
zcc.qwer@gmail.com

Chun Yu‡
Department of Computer Science and Technology, BNRist, College of AI, Tsinghua University
Beijing, China
chunyu@tsinghua.edu.cn

Yuanchun Shi
Department of Computer Science and Technology, BNRist, Tsinghua University
Beijing, China
Qinghai University
Xining, China
shiyc@tsinghua.edu.cn

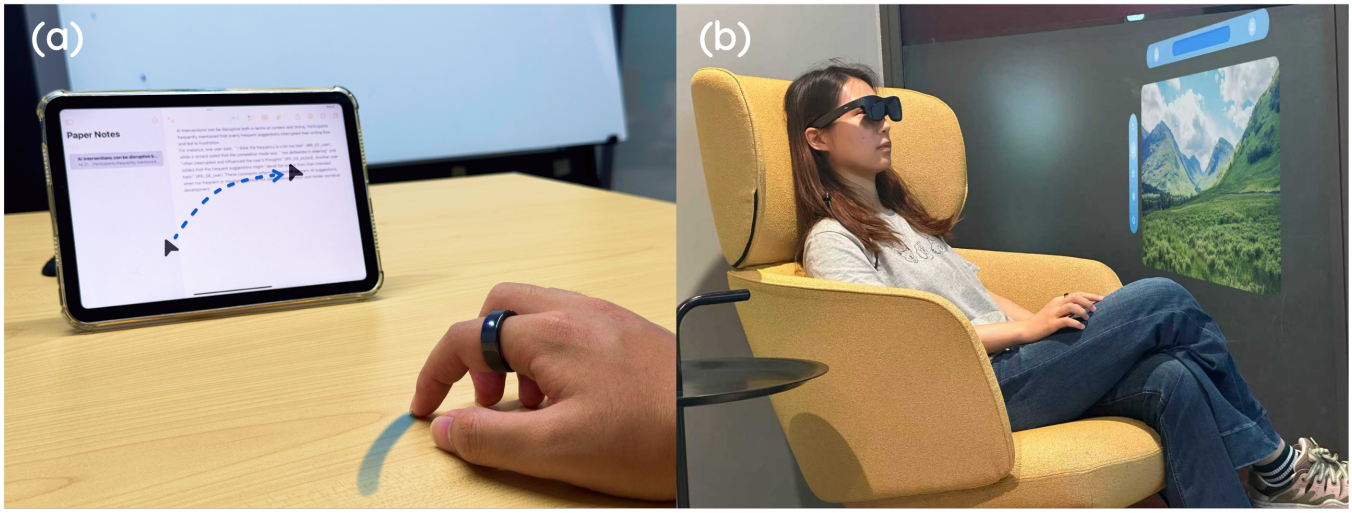


Figure 1: The demonstration of TraceRing. (a) TraceRing supports touchpad-like pointing with a single IMU Ring through Personalized Learning. (b) TraceRing can provide ubiquitous and precise pointing input for VR, AR, and large-display interaction scenarios.

Abstract

Achieving touchpad-like pointing with a single IMU ring is highly desirable for portable and wearable interaction, yet challenging due to incomplete motion data and significant user variability. We present TraceRing, a finger-worn IMU system that enables precise two-dimensional cursor control. To address the limitations of generic end-to-end models, we propose a personalized training framework that learns user-specific representations through joint multi-task and contrastive learning, while dynamically selecting the most suitable expert model. This approach enables personalization without requiring per-user fine-tuning, and reduces velocity prediction error by 33.9% over state-of-the-art baselines. Furthermore, a real-time study shows it delivers speed and accuracy far exceeding those of AirMouse (2.26s v.s. 3.01s in average task completion time). These results demonstrate TraceRing as a portable and comfortable alternative for mobile computing and AR interaction applications.

CCS Concepts

• **Human-centered computing** → **Mobile devices**; **Pointing devices**; **Pointing**.

Keywords

Pointing Technique, IMU, Ring Interaction, Personalized Learning, Touch Interface, Wearable Devices

ACM Reference Format:

Zhe He, Weinan Shi, Zixuan Wang, Suya Wu, Xiyuan Shen, Chengchi Zhou, Chun Yu, and Yuanchun Shi. 2026. TraceRing: Touchpad-like Pointing with a Single IMU Ring through Personalized Learning. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3772318.3790463

1 Introduction

Pointing is one of the most fundamental interaction techniques in Human-Computer Interaction (HCI). With the widespread adoption of mice and touchpads, people are already accustomed to precise 2D GUI control. However, as display technologies diversify and mobile computing advances, devices such as VR/AR headsets, large screens, and tablets create a growing need for precise and ubiquitous 2D pointing.

Researchers have explored alternatives, including remote controllers [36, 38], cameras [12, 53], eye-tracking [10, 56], and head-tracking [39, 54]. However, these methods are often limited by hardware requirements and fail to generalize across diverse scenarios. In contrast, wearable-based approaches avoid environmental hardware constraints and have attracted broad interest. Prior studies have demonstrated 2D pointing using wristbands [22, 35], watches [24, 61], rings [26, 42, 47], or finger-worn devices [59] equipped with sEMG [22, 35], IMU [24, 26, 42, 47, 61], infrared proximity sensor [26], or micro RGB camera [59]. Among these, IMUs stand out for their compact size, low power consumption, and affordability, making them promising candidates for always-available input.

Most IMU-based pointing systems adopt a ring form factor, as IMUs measure finger motion most accurately when placed close to the finger. However, prior work often relies on additional sensors [26] or multiple rings [47], while single-IMU solutions typically yield suboptimal performance [42, 47]. To advance toward a truly ubiquitous solution, we investigate why a single IMU struggles to reconstruct fingertip motion and propose a method to overcome this limitation.

We introduce TraceRing, a system that achieves touchpad-like pointing with a single IMU ring through personalized learning. To investigate the problem, we first collected a dataset from 50 participants with precise annotations, including synchronized IMU, pressure pad, and motion capture data. Our analysis revealed a key challenge: the presence of many "one-to-many" mappings, where similar finger-root motions (captured by IMU and motion capture) can correspond to different fingertip motions across individuals. This inconsistency prevents the learning of a universal model and highlights the need for personalization.

Unlike prior personalization methods that require additional calibration data and costly fine-tuning, TraceRing employs a joint learning framework that achieves personalization without explicit calibration. The architecture consists of two main components: a Task Encoder (TCN + LSTM), trained with supervised learning to capture task-related embeddings, and a User Encoder (BiLSTM + temporal attention) trained with contrastive learning to extract user-specific embeddings. These embeddings are combined through a Mixture-of-Experts (MoE) module, where multiple MLP experts map task embeddings to fingertip motion, and the user embeddings dynamically weight their outputs. This design provides efficient, low-overhead personalization while maintaining good performance.

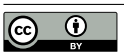
Evaluation on our dataset shows that TraceRing reduces error in velocity prediction by 33.9% over the previous state of the art. The User Encoder can also distinguish users with as little as 15 seconds of data and generalize to unseen participants. To evaluate usability, we conducted a real-time Fitts' Law experiment, comparing TraceRing with a commercial IMU air mouse and a touchpad baseline. Results show that TraceRing (MT = 2.26s) significantly outperformed the airmouse (MT = 3.01s). While participants noted that TraceRing's accuracy and latency still lagged slightly behind touchpads, most expressed a strong willingness to adopt it in practice, citing its portability and ease of use.

In summary, our contributions are threefold:

(1) We introduce TraceRing, the first single-IMU ring system that delivers touchpad-like pointing through personalized learning, enabling reliable pointing on flat surfaces for ubiquitous interaction.

(2) We systematically analyze why prior single-IMU approaches underperform and propose a novel multi-task personalization framework that adapts to individual users without calibration or fine-tuning.

*Corresponding author.
†Also with Department of Computer Science and Technology, Tsinghua University.
‡Also with Key Laboratory of Pervasive Computing, Ministry of Education.



CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3790463

(3) We demonstrate TraceRing's practical usability through a real-time experiment, showing high performance, rapid learnability, good generalization across users and strong user acceptance.

2 Related Work

2.1 Pointing Techniques

The pursuit of natural and intuitive pointing interaction for applications in AR, VR, and large-screen displays has led to the establishment of three dominant technical paradigms. These are primarily realized through vision-based systems, which leverage cameras to capture hand, face, or eye motion for gesture recognition and tracking [9, 10, 23, 56, 64]; surface electromyography (sEMG) sensors, which detect muscle activity to infer motion intentions [11, 21, 51]; and Inertial Measurement Units (IMUs), which track hand or head posture using accelerometers and gyroscopes [2, 13, 58]. However, these modalities present inherent trade-offs that hinder their universal applicability. Vision-based approaches are fundamentally constrained by high energy consumption and sensitivity to ambient lighting [28]. sEMG systems grapple with challenges in signal fidelity and are particularly susceptible to muscle fatigue, compromising their reliability for sustained use [8, 27, 46].

To overcome these challenges, multi-sensor fusion Han et al. [15], Liu et al. [35] and algorithmic improvements [22, 65] have been explored. Han et al. [15] combined sEMG and IMU in a custom wristband to achieve high-precision micro-gesture recognition, though dataset representativeness was limited. Kaifosh et al. [22] proposed a non-invasive sEMG neuromuscular interface with transfer learning and personalized tuning for cross-user generalization, yet device power and computational cost constraints still existed. MouseRing [47] employed one or two IMU rings for fingertip sliding tracking, with the dual-ring system being more precise but less comfortable, and the single-ring system prioritizing comfort at the cost of accuracy.

Although pointing techniques represented by the mouse and touchpad are already highly mature, the field continues to explore more unobtrusive and convenient solutions that incur lower computational cost while maintaining accuracy, in order to support emerging scenarios such as AR. In this context, TraceRing is the first to achieve accurate pointing using only a single IMU ring, thereby resolving the critical trade-off between precision and wearability. This has not been accomplished in prior work and has substantial practical value.

2.2 Ring Based Input

Ring-based interfaces have emerged as a promising form factor for subtle and always-available input, leading to an exhaustive exploration of sensing technologies and interaction paradigms. However, a significant category of these designs sacrifices the core value of portability by requiring auxiliary wrist-worn devices to achieve high performance. For instance, systems like PicoRing [33], AuraRing [44], and Finexus [7] leverage powerful electromagnetic sensing for continuous tracking, but their functionality is tethered to a companion wristband. This dependency extends to gesture-focused designs like Thumb-In-Motion [1] and hybrid systems like Ring-watch [57]. While effective, these multi-component systems compromise the unobtrusive, self-contained interaction that defines a ring's value proposition, thereby limiting their practicality in daily use.

In pursuit of true portability, another line of research has focused on developing standalone rings that operate without auxiliary hardware. However, this approach has often led to a new set of compromises across interaction fidelity and user generalization. Some designs incorporate novel sensing modalities that introduce new vulnerabilities: Ring-a-Pose's [62] acoustic sensing is crippled by poor battery life (≈1.75 hours), and TouchRing's [50] capacitive sensing struggles with stable gesture recognition. Other systems fall short in interaction fidelity; [18] is hampered by high latency from external server communication. LightRing [26] imposes strict postural constraints that limit the user's range of motion. Ultimately, and perhaps most critically, achieving user generalization remains a persistent hurdle. Systems like SkinRing [16] exemplify this challenge, being susceptible to infrared interference and necessitating burdensome, user-specific retraining, which undermines a seamless "pick-up-and-use" experience. FlowRing[6] combines an optical flow sensor, a skin-contact microphone, and an IMU to simultaneously provide 2D pointing and micro-gesture recognition. Although it can achieve reasonably accurate pointing, it imposes strict requirements on device size and battery capacity, because the optical flow chip and its associated regulators consume 30 times more power than the IMU and also occupy substantially more space. In addition, the micro-gesture recognition in this work exhibits limited generalizability.

TraceRing is designed to address these limitations while explicitly targeting touchpad-like pointing using only a single IMU ring. As a truly standalone device, it avoids the portability constraints of auxiliary-dependent systems and high demand of power consumption. At the same time, it tackles the interaction and generalization challenges faced by prior standalone rings. TraceRing demonstrates that high-fidelity pointing can be achieved with a single IMU ring, highlighting the practical value of this form factor for sensor-based interaction.

2.3 Personalized HAR

In the context of IMU-based touch input recognition on smart rings, this task falls within fine-grained human activity recognition (fine-grained HAR), where individual differences in users' finger motion patterns are particularly pronounced [55]. To address such personalization challenges, a common strategy is to train a global model on a diverse dataset and fine-tune it with a small amount of user-specific data [31, 45]. Other methods introduce decoupled architectures, personalizing only specific model layers (e.g., the classification head) to reduce the adaptation overhead [32, 41, 48]. While effective, these approaches often require a non-trivial amount of personal data for fine-tuning and may not fully leverage the knowledge from a cohort of existing users who share similar motion patterns with the new user. This limits their data efficiency and generalization potential, especially in a cold-start scenario for new users.

Some works have explored clustering-based methods to leverage shared data characteristics better. These schemes group users with similar data distributions during the training phase and train

cluster-specific models [5, 14, 43, 63]. However, these static clustering frameworks are primarily designed to optimize model performance for a known set of users. They often lack a dedicated, efficient mechanism for dynamically assigning a new, unseen user to an appropriate cluster post-deployment without significant recomputation. More advanced systems have begun to address the challenge of new users. Recent work in federated learning for HAR, such as FedCHAR, has explicitly proposed extensions that dynamically adapt to new participants [34]. While these approaches validate the importance of dynamic adaptation, they often focus on the server-side logic of re-clustering or evolving the model pool. The challenge of creating a highly efficient, lightweight client-side mechanism for a new user to instantly match with the most suitable pre-trained model remains an open area. In contrast, our proposed method addresses these gaps by introducing a novel, two-stage personalization strategy. First, we leverage contrastive learning, a technique proven effective in learning robust representations from time-series sensor data [19, 25], to generate highly discriminative embeddings of finger motion patterns. Based on these embeddings, we pre-train and cluster a set of specialized 'recognition heads'. For a new user, instead of requiring extensive fine-tuning or server-side reclustering, our system employs a lightweight dynamic matching mechanism to select the most compatible recognition head. This approach achieves efficient and effective personalization for smart ring-based touch input, significantly lowering the barrier for user adaptation.

3 Problem Formulation

TraceRing aims to enable 2D pointing input on arbitrary surfaces using a single IMU ring, providing users with a precise input experience similar to a touchpad. Since most users are already familiar with touchpad usage, we aim to allow such experience to transfer directly, enabling seamless adoption of TraceRing without additional learning effort. To achieve this, we propose three key questions:

KQ1: How can we formally define the problem of reproducing the touchpad input experience?

Although a touchpad can accurately detect the absolute position of the fingertip on a surface, cursor controlling is essentially realized by mapping short-term relative displacements of the contact point to cursor movements, which can be regarded as a velocity-to-velocity mapping through a CD gain [3]. Therefore, if we can reconstruct the instantaneous velocity of the fingertip on the surface, we can reproduce the cursor control effect of a touchpad. Another reason for reconstructing fingertip velocity rather than absolute position is that most wearable sensors cannot perceive a fixed absolute coordinate system.

Accordingly, we define the problem as follows: given sensor observations X, prior information z, and model parameters θ, our objective is to estimate the velocity V. Based on maximum a posteriori (MAP) estimation, this process can be formulated as:

$$\begin{aligned}\hat{V} &= \arg\max_{V} p(V \mid X, z, \theta) \\ &= \arg\max_{V} \log p(X \mid V, \theta) + \log p(z \mid V, \theta) + \log p(V \mid \theta)\end{aligned} \tag{1}$$

The **first** term indicates that the velocity V is the most probable solution inferred from the sensor observations X. In the **second** term, we summarize all prior information as a latent variable z, whose meaning varies across different works. For approaches with high sensing performance (often at the cost of wearing comfort), little prior information is needed to obtain accurate velocity estimation. For example, Yang et al. [59] employed an optical sensor similar to a mouse, achieving high tracking accuracy but offering a less comfortable wearing experience. The **third** term represents assumptions about the velocity itself, such as its range, smoothness, and other constraints.

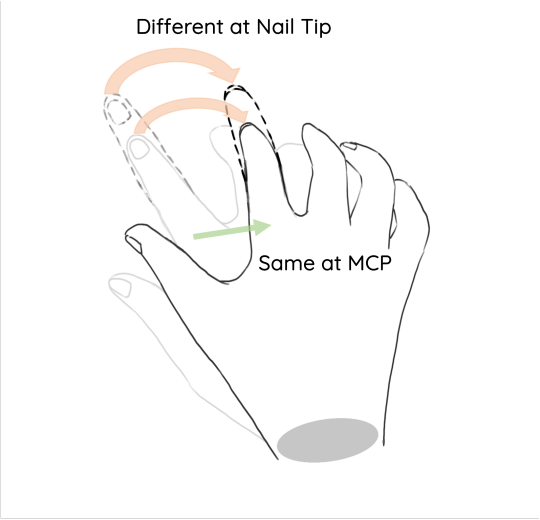


Figure 2: This illustration depicts how identical motion at the MCP joint can result in different movements at the nail tip when physiological parameters, such as finger length, vary.

KQ2: Is real-time data from a single IMU ring sufficient to reproduce the touchpad input experience?

We first review a recent related work, MouseRing [47]. It employs two IMU rings (worn on the proximal phalanx and intermediate phalanx, respectively) to enable a touchpad-like input experience. In this work, X represents the observations from the two IMUs, while z denotes the physical modeling priors (e.g., the coplanarity of the velocities at finger joints). Although the two IMUs provide much richer observations than a single IMU, strong physical modeling priors are still introduced to achieve an input experience close to a touchpad. The study also evaluated a single-ring setup, but its performance was significantly inferior to the dual-ring configuration.

This raises a question: are the observations from a single IMU sufficient for this task? Since the IMU is worn at the finger's base, its distance from the fingertip makes it challenging to model the entire finger accurately, as with the dual-IMU setup. Moreover, due to differences in individual physiology, the same fingertip velocity V may correspond to different IMU observations X at the finger base. Conversely, the same IMU observation X at the finger base may correspond to different fingertip velocities V, as shown in Fig. 2. The hypotheses have been empirically validated in Sec. 4.5.

Therefore, we argue that under the assumptions of prior work, reproducing the touchpad input experience with a single IMU cannot be accomplished using their original approach.

KQ3: How can priors be modified to overcome the information limits of a single IMU?

Based on Equation 1, we cannot alter the prior assumptions on the velocity V itself, and our observations X are weaker compared to related work. Therefore, we require a stronger prior z to make the problem solvable. To this end, we propose a personalization-based approach. Specifically, we use a short segment of data to distinguish between users, and then select different priors z and parameters θ according to individual characteristics, thereby improving the robustness and precision of velocity prediction. In other words, personalization-based priors z are introduced to compensate for the limitations of the observations X, reducing the uncertainty in the mapping from X to V. We validate the feasibility of this approach experimentally in Sec. 5.3.

4 Data Collection and Analysis

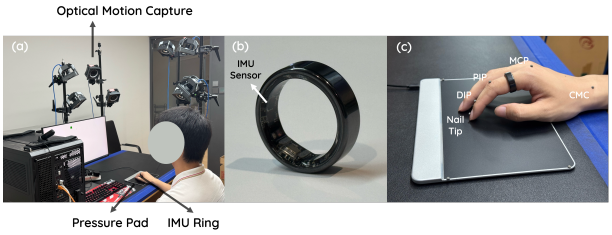


Figure 3: We collected a finely annotated touchpad interaction dataset using optical motion capture, a pressure plate, and an IMU ring. (a) The data collection system. (b) The IMU ring used in the experiment. (c) We placed markers on different joints of the index finger to accurately track the movement of the finger.

To verify our hypotheses in KQ2 and KQ3 and to build the complete system, we first conducted a large-scale data collection experiment. We aimed to comprehensively record the entire process of user-touchpad interaction to enable detailed analysis and methodological design.

4.1 Apparatus

To ensure that user behavior was as close to real usage as possible, we did not employ a wired setup. Instead, we used a wireless smart ring equipped with a six-axis IMU to collect data (see Fig. 3). We worked with the manufacturer to custom-build a smart ring inspired by the τ-ring [49]. Compared with the τ-ring, our design removes the PPG sensors and adds a capacitive touch sensor. The ring is based on the ICM-42688 IMU sensor and transmits data to the computer via BLE at a sampling rate of 200 fps. During the experiment, participants were instructed to wear the device in a fixed orientation so that the IMU axes remained consistent across all trials. When the finger rested flat on the desk, the positive directions of the x, y, and z axes corresponded to forward, rightward, and downward, respectively. We kept the wearing orientation consistent with subsequent experiments to ensure data consistency. In practical deployments, the orientation can be estimated from gravity and the sensor axes rotated accordingly, or handled via data augmentation. For simplicity, we did not implement these steps, as we consider them outside the scope of this work and a known, addressable engineering issue. We acknowledge this as a limitation of our work.

To capture the ground-truth displacement of the fingertip, we used a Sensel Morph[1] pressure pad to record touch positions, sampled at approximately 240 fps. We also implemented a simple cursor control scheme on top of it to provide users with feedback (Fig. 3(a)). We assumed that there was only one touchpoint at any given time. We directly mapped the displacement between consecutive touch positions directly to the cursor's movement direction and distance (without CD gain). If the interval between touch-down and lift-off was shorter than 0.5 s, we regarded it as a click event at the corresponding location.

In addition, to obtain full-hand pose data throughout the interaction, we employed an OptiTrack optical motion capture system. Five 2 mm infrared reflective markers were affixed to the carpals (CMC), metacarpophalangeal joint (MCP), proximal interphalangeal joint (PIP), distal interphalangeal joint (DIP), and the nail tip (NT) (see Fig. 4). 8 surrounding cameras were used to track the 3D absolute positions with sub-millimeter accuracy, at a sampling rate of 120 fps.

During the experiment, users received visual feedback of the cursor movements on a 23.8-inch display with a resolution of 1920×1080, positioned approximately 40 cm in front of them.

4.2 Participants

We recruited 50 participants from a university campus and its surrounding community (21 male, 29 female, aged 16–38, M = 25.78). All participants were right-handed and had no excessively long fingernails. The diameter of each participant's finger was smaller than the inner diameter of the ring; for users with skinny fingers, we applied nano tape inside the ring to ensure a secure fit.

4.3 Design and Procedure

Since we aimed for users' behavior to be as consistent as possible with real touchpad usage, we did not design a wide variety of tasks; instead, we employed a single Fitts' Law task. In the experiment, participants controlled a cursor by moving their finger on the pressure pad and performed single clicks with finger taps. In each trial, a square target randomly appeared on the interface. Participants were required to move the cursor into the target region using the pressure pad and then click. Upon a successful click, the next square target appeared.

Each trial consisted of 20 targets. The target size was set to one of three values—20 px, 50 px, or 100 px—which were alternated in sequence. Each participant completed 8 blocks of 24 trials, with each block containing all three target sizes. Participants took at least a 30-second break between blocks; some slower participants completed only 6 blocks. The entire experiment lasted approximately 30 minutes, and each participant received 10 dollars as reward.

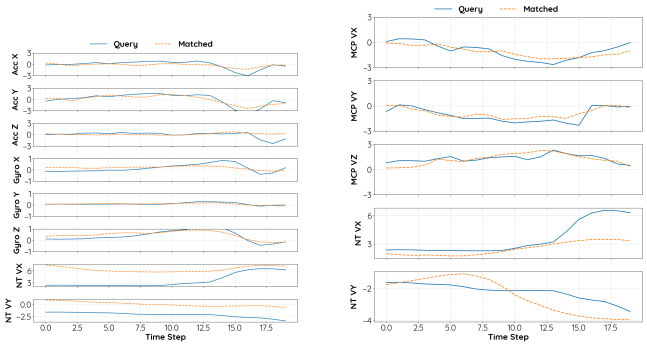
[1] https://morph.sensel.com/

4.4 Data Processing

Since our study involves multimodal sensor data with different sampling rates, the first step was to align them. The pressure pad and the optical motion capture system were connected to the computer via wired connections, both using the PC's local clock. The IMU ring, however, transmitted data via BLE and relied on its onboard timestamps. To synchronize, we first calculated the precise sampling frequency of the IMU (relative to PC time, approximately 199.84 fps). We manually calibrated the start time of each recording session to ensure that the IMU's click peaks aligned with the touch signals detected by the pressure pad, thereby converting the ring's local timestamps into PC timestamps.

Verification confirmed that the movement data from the pressure pad were temporally consistent with the nail tip trajectories captured by the motion capture system, and that the temporal offset between the IMU click peaks and the pressure pad's touch signals was generally less than 0.05 seconds (i.e., aligned at 200 fps). In this way, temporal synchronization across modalities was achieved. After synchronization, we trimmed the data to the shortest time span among the three modalities and applied linear interpolation to align the pressure pad and motion capture data with the IMU data at every timestamp.

4.5 Preliminary Data Exploration



(a) The closest IMU segment comparison.

(b) The closest optical motion capture segment comparison.

Figure 4: Using the data segment at MCP as the query, compare it with the matched segment that is close in distance. Even if the data at MCP appear almost identical, the motion data of the fingertips shows significant differences.

To verify the hypothesis proposed in Section 3, that the IMU data from a single ring is insufficient to reconstruct finger movements on a touchpad, we conducted a preliminary data exploration on the dataset.

4.5.1 The mapping from MetaCarpoPhalangeal(MCP) to Nail Tip. We first aim to illustrate the assumption introduced in Section 3: the data exhibit a "one-to-many" relationship. Specifically, the same motion at the MCP (reflected in the IMU signals and the optical marker positions) may correspond to different motions at the Nail Tip (reflected in the touchpad signals). To examine this, we randomly sampled several 20-frame (0.1 s) segments from the dataset, ensuring that the index finger remained in contact with the touchpad. For each sampled segment, we then identified both the most similar IMU segment and the most similar MCP-marker motion segment in the dataset. We measured the similarity between two segments using Euclidean distance, and employed the method in [37] for acceleration. Since the ranges of acceleration and angular velocity differ for IMU data, we first scaled the angular velocity by a factor of three before matching.

We present one representative result in Fig. 4, with additional results in Appendix A. In the figure, IMU or motion capture data at the MCP are used as the query, and the most similar segment from the dataset is retrieved, denoted as matched. The two segments are nearly identical regarding MCP data; however, their corresponding Nail Tip motions (bottom two rows) differ substantially. This demonstrates that relying solely on a single IMU placed at the MCP is insufficient to directly reconstruct the touchpad motion at the Nail Tip, which also helps explain why the single-ring configuration in [47] performed poorly.

4.5.2 Wrist Position. We also analyzed the wrist positions of participants during input. Although the experiment did not impose any constraints on hand posture, most participants naturally chose to keep their wrist fixed in one position and relied solely on the index finger to accomplish the target selection tasks. Statistical analysis shows that in 84.7% of the trials, the displacement of the wrist marker was less than 1 cm, indicating that fixing the wrist while using a touchpad is a natural and comfortable choice. Moreover, since this posture partially reduces the degrees of freedom in finger movement, we adopted it in subsequent experiments to facilitate learning of the motion mapping between the MCP and the Nail Tip.

5 Method

Next, based on the characteristics of the task of achieving touchpad-like pointing with a single IMU, we designed a personalized learning–based algorithm and evaluated its performance on the dataset. Thus, we validated the hypothesis from Section 3: that incorporating personalized priors can overcome the limitations of single-IMU data.

5.1 Algorithm Pipeline

Figure 5 illustrates our algorithmic pipeline. We first perform orientation estimation and gravity compensation on the raw IMU input, and then feed the processed signals into two branches: a task encoder and a user encoder. The task encoder, responsible for extracting task-related information (i.e., velocity reconstruction), comprises a TCN module followed by an LSTM and outputs a task embedding. The user encoder, responsible for extracting user-specific characteristics, consists of a bidirectional LSTM and a temporal attention mechanism; it converts a short segment of a user's input into a user embedding. The task embedding is passed through MLP-based Expert module(s), and the Expert outputs are combined by a weighted sum where the weights are produced by applying a linear layer to the user embedding; the result is the final predicted velocities along the x and y axes. We replaced the commonly used segmented regression training paradigm with a

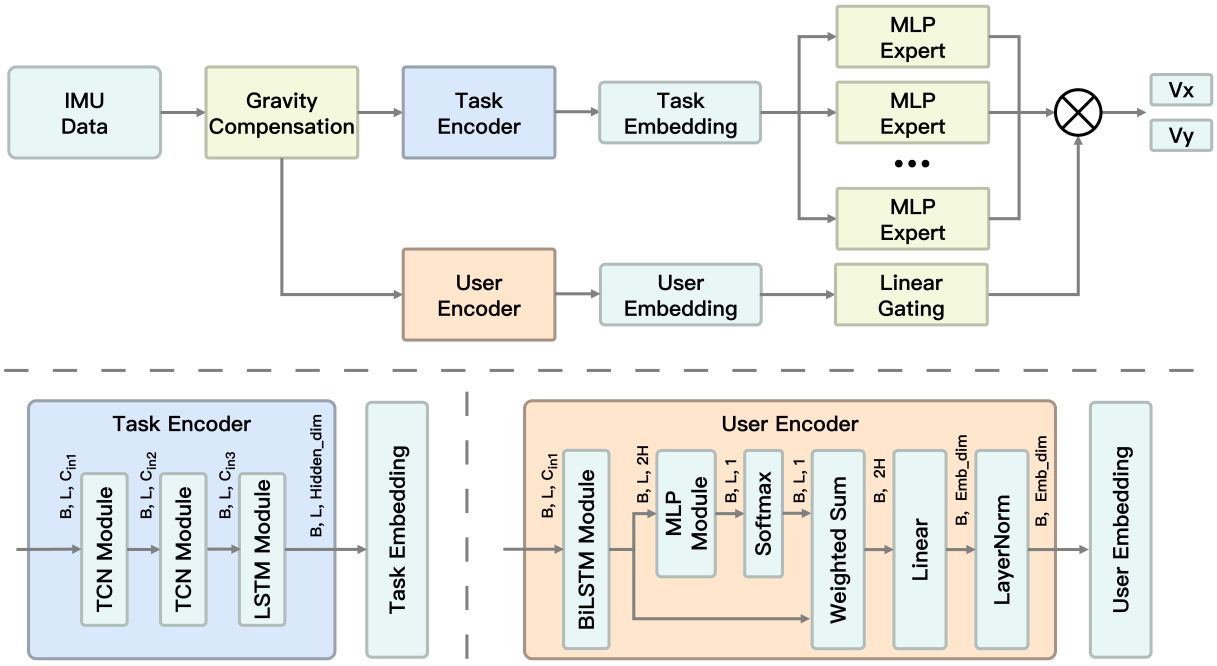


Figure 5: Algorithm Pipeline for TraceRing.

seq2seq approach to enable the model to capture richer task-related temporal information.

Below, we describe each component in detail and present the training and inference procedures.

5.1.1 Gravity Compensation. Since the gravity component typically dominates the accelerometer readings but is irrelevant to our task, we remove it to improve the stability of model inference. To eliminate gravity, we first estimate the ring's orientation using the VQF algorithm [29], which accurately computes the real-time IMU orientation from six-dimensional accelerometer and gyroscope inputs. With the estimated orientation, we project the gravity vector onto each axis of the IMU accelerometer and subtract it accordingly. The resulting signals represent the linear acceleration and angular velocity in the IMU's local coordinate system. Using these gravity-compensated signals not only enhances inference stability but also makes the algorithm naturally applicable to touch surfaces at arbitrary orientations.

5.1.2 Task Encoder. The goal of the Task Encoder is to capture as much fingertip velocity–related information from the input as possible. To this end, we design a TCN-LSTM architecture, where the TCN module extracts short-term temporal patterns from local input segments, and the LSTM module models sequential dependencies over the entire input sequence, producing an embedding of size hidden_dim at each time step. We adopt TCN because its large receptive field and causal convolution ensure accurate prediction coverage. The LSTM complements this by capturing temporal dependencies, particularly emphasizing the most recent inputs. Since instantaneous fingertip velocity is mainly determined by short-term motion history and longer-term user-specific dynamics, this architecture is well-suited to capture both aspects effectively.

5.1.3 User Encoder. The goal of the User Encoder is to capture as much user-specific information as possible, to guide the selection of the most suitable Expert for each individual. We first apply a bi-directional LSTM, which encodes contextual relationships across time and produces features for each timestep. These features are then passed through an MLP followed by a softmax layer, whose outputs are used to compute a weighted sum over the original features. This temporal attention mechanism allows the model to emphasize user traits revealed in critical actions (e.g., pressing down or lifting off). Finally, the result is passed through a Linear layer and LayerNorm to produce the user embedding. Although this design does not inherently constrain the input length, in our experiments we fix the sequence length L to ensure stable and effective user modeling.

5.1.4 MoE Module. To achieve personalization, we adopt a Mixture of Experts (MoE) architecture, where multiple MLPs serve as Experts to map the task embedding into fingertip velocities. Instead of including the entire Task Encoder within each Expert, we only use the prediction heads as Experts. This design significantly reduces the parameter count and lowers both training and inference costs. The user embedding is passed through a linear gating network, which produces weights to combine the outputs of the MLP Experts into the final predicted fingertip velocities (Vx and Vy). In practice, computational efficiency can be further improved by applying a top-k selection strategy, activating only a subset of the Experts during inference.

5.2 Model Training

5.2.1 Training Process. During training, to ensure the model captures sufficient historical information, we segment the data into

sequences of length 3000 frames (i.e., 15 seconds). Since the index finger does not remain on the plane for the entire segment, we also record a corresponding mask, assigning a value of 1 to timesteps where the finger is in contact with the plane. After passing the input through the model, we obtain the predicted fingertip velocities along the x and y axes(v_t) at each timestep. Multiplying these predictions by the mask(m_t), we compute the MSE loss with respect to the ground truth and normalize by the number of valid timesteps in the mask, yielding the velocity loss $\mathcal{L}_{\text{vel}}$:

$$\mathcal{L}_{\text{vel}} = \frac{\sum_{t=1}^{T} m_t \left((\hat{v}_t^x - v_t^x)^2 + (\hat{v}_t^y - v_t^y)^2 \right)}{\sum_{t=1}^{T} m_t}, \tag{2}$$

To ensure that the user embedding effectively distinguishes between different users, we optimize it using a contrastive learning approach. Specifically, for each batch, we sample N users, and for each user, we select M data segments. After passing these segments through the model and normalizing, we obtain the user embedding z_i, with each embedding corresponding to a user u_i. The similarity between samples, s_{ij}, is then defined as:

$$s_{ij} = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\tau} \tag{3}$$

Here, τ is the temperature coefficient, which we empirically set to 0.07. Denoting the set of positive samples as $P(i) = \{j | u_j = u_i\}$, the contrastive loss $\mathcal{L}_{\text{ctr}}$ is defined as:

$$\mathcal{L}_{\text{ctr}} = -\frac{1}{N \times M} \sum_{i=1}^{N \times M} \sum_{j \in \mathcal{P}(i)} \frac{1}{|\mathcal{P}(i)|} \log \frac{\exp(s_{ij})}{\sum_{k=1}^{N \times M} \exp(s_{ik})} \tag{4}$$

This loss encourages embeddings from the same user to be closer. However, relying solely on the contrastive loss would limit the model to distinguishing only among users in the training set, reducing its ability to generalize to unseen users. To address this, we use K learnable prototypes and compute the similarity between each sample and all prototypes, denoted as s_{ik}^{proto}, which is defined as:

$$s_{ik}^{proto} = \frac{\mathbf{z}_i^\top \mathbf{p}_k}{\tau} \tag{5}$$

We identify the closest prototype to each sample and treat it as its corresponding category. A cross-entropy loss is then applied to enforce this assignment, defined as:

$$\mathcal{L}_{\text{proto}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(s_{iy_i}^{\text{proto}})}{\sum_{k=1}^{K} \exp(s_{ik}^{\text{proto}})} \tag{6}$$

Here, $y_i = \arg\max_k s_{ik}^{\text{proto}}$ denotes the index of the closest prototype to the sample. Finally, the overall loss is expressed as a linear combination of the three components:

$$\mathcal{L} = \mathcal{L}_{\text{vel}} + \lambda_{\text{ctr}} \mathcal{L}_{\text{ctr}} + \lambda_{\text{proto}} \mathcal{L}_{\text{proto}} \tag{7}$$

In our experiments, we empirically set the number of prototypes to 16, $\lambda_{\text{ctr}} = 0.1$, and $\lambda_{\text{proto}} = 1$.

Table 1: Performance comparison of different models. These errors are measured in cm^2/s^2.

Model	VError	XError	YError
MouseRing	2.24	2.22	2.27
TraceRing w/o User Encoder	1.73	1.76	1.70
TraceRing	**1.48**	**1.51**	**1.45**
TraceRing-2-Experts	1.58	1.56	1.61
TraceRing-4-Experts	1.55	1.57	1.53
TraceRing-6-Experts	1.62	1.65	1.60
TraceRing-8-Experts	**1.48**	**1.51**	**1.45**
TraceRing-10-Experts	1.58	1.53	1.63

5.2.2 Training Settings. In our experiments, the Task Encoder consists of two TCN modules with a kernel size of 3 and channel sizes of 64 and 128, respectively. The hidden dimension of the LSTM module is set to 64, and the user embedding dimension is also set to 64. All of these parameters were selected empirically. The training was conducted using the AdamW optimizer, with a fixed learning rate of 0.001 and a weight decay of 0.0001. We trained our model on a server equipped with an NVIDIA RTX 4090 GPU, using a batch size of 64 for a total of 2000 epochs, and reported the best performance on the test set.

5.3 Evaluation on Dataset

We first evaluate the model's performance on the dataset. For the 50-user dataset, we adopted a 5-fold cross-validation strategy: all users were divided into 5 groups of 10, each serving as the test set once while the remaining groups formed the training set. The final results were obtained by averaging across all folds.

5.3.1 Velocity Prediction Performance. As shown in Table 1, we evaluated the performance of different models on our dataset. We measured the accuracy of velocity reconstruction using the mean squared error (MSE) between the predicted and ground truth fingertip velocities at each frame. Additionally, we separately report the MSE for the x- and y-axis velocities. We used the single-ring configuration of MouseRing [47] as our baseline. We also experimented with different numbers of Experts. The results demonstrate that, compared to the previous state-of-the-art MouseRing, our approach achieves a significant improvement of 33.9% in velocity prediction error (Table 1), greatly enhancing the user experience. Moreover, testing with varying numbers of Experts shows that the differences among the other configurations are minor. For the subsequent experiments, we adopted the best-performing configuration with 8 Experts. An ablation study also shows that removing the user encoder (retaining only the task encoder) leads to a substantial performance drop (1.48 to 1.73).

5.3.2 User Embedding Visualization. To evaluate the effectiveness of the User Encoder, we randomly selected a fold and visualized all embeddings from its training and test sets using t-SNE, as shown in Fig. 6. In the figure, larger dots with black borders represent test set data, while lighter dots without borders represent training set data.

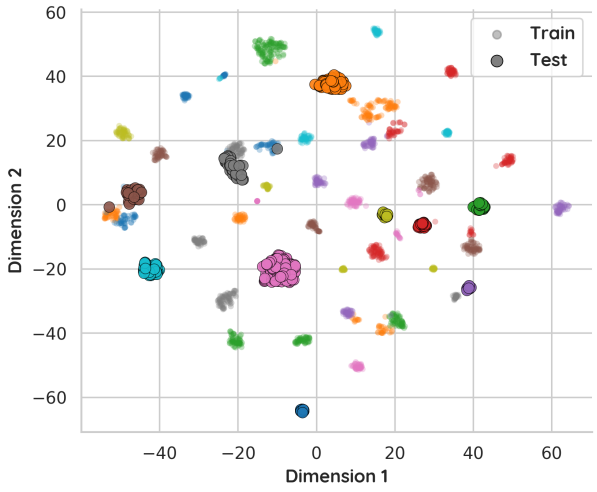


Figure 6: TSNE results of the user embeddings on both train and test dataset. Different colors represent different users. Large circles with borders indicate test set results, while small circles without borders indicate training set results.

It can be observed that embeddings from the training set are completely separable, indicating that the model accurately captures inter-user differences. Even for test set data from users unseen during training, most embeddings from the same user form tight clusters, while embeddings from different users remain distinguishable. Moreover, the test set embeddings do not form a single compact cluster but are distributed across the training set's distribution. Despite occasional outliers, these results suggest that the model can distinguish between different users and identify users with similar characteristics.

5.3.3 Temporal Attention Visualization. In the User Encoder, we employ a weighted sum to emphasize timesteps that better reflect inter-user differences. To illustrate this, we visualize the weights, as shown in Fig. 7. Each weight peak almost exactly precedes an acceleration spike, rather than aligning with the spike itself. For angular velocity, the peaks typically occur at the center of an up–down–up oscillation pattern.

In our dataset, nearly every acceleration spike corresponds to a finger pressing action during a click. This indicates that the model places more emphasis on the data immediately before and after a click, since these segments involve larger finger movements that more clearly reveal user-specific differences. In contrast, the acceleration spike at the exact moment of clicking contains excessive noise, which explains why the corresponding weight is relatively low.

5.4 Real-time Usage

The previous subsection primarily evaluated the algorithm's performance on the dataset. However, in practical use, it is not sufficient to merely reconstruct fingertip velocity. We also need touch detection to determine when to move and when to stop, filtering mechanisms and CD gain to make the cursor behavior closer to that of a conventional touchpad. In addition, we must design an efficient strategy for computing the user embedding—one that avoids recalculating it at every frame, thereby reducing computational overhead while maintaining good performance.

5.4.1 Touch Detection. We first crop the raw data into segments and use the pressure pad readings to label each IMU frame as contact (1) or no contact (0). We then treat each transition from 0 to 1 as a "Press" event and each transition from 1 to 0 as a "Lift" event.. We adopted the four-class event framework from [17], classifying each data window into one of four states: **Contact**, **In-air**, **Press**, and **Lift**. Including stable states (Contact and In-air) enhances the model's robustness in distinguishing between transient transitions and stable conditions, thereby effectively reducing false positives.

Accordingly, based on these states, we designed an event-driven windowing and sampling strategy. For "Press" events, we extract an asymmetric window (15 timesteps before the event, 5 after) to capture the preparatory motion. For "Lift" events, we extract a symmetric window centered on the event point. This design is crucial to prevent the sampling window of a "Lift" event from being contaminated by data from a preceding "Press" event, especially in rapid tapping scenarios. Conversely, since our dataset contains no instances of 'Lift-then-immediately-Press,' employing an asymmetric window for 'Press' events is safe. This design also allows the system to complete detection just 5 timesteps after an event occurs, significantly reducing interaction latency. For the stable states "Contact" and "In-air," we sample windows randomly from their corresponding long-duration segments to ensure sample diversity. Through this methodology, we constructed a clearly labeled dataset focused on both critical transitions and stable states.

On this dataset, we utilized the InceptionTime model [20] for classification. We trained the model using the tsai library [40], with 6-channel IMU data as input and the four event classes as output. The number of initial filters (`nf`) was set to 16. We employed a batch size of 1024 and the one-cycle training policy for 15 epochs, with a maximum learning rate of 5e-3. The final model achieved an accuracy of 97.1% on the test set.

To implement the click functionality and enable the TraceRing system to be used like a conventional pointing device, we designed a threshold-based algorithm. When a "Lift" event is detected, if the elapsed time since the most recent "Press" event is less than 100 ms and the cursor displacement during this interval is less than 20 pixels, the interaction is classified as a click. The 20 px constraint prevents rapid (each shorter than 100 ms), consecutive small cursor adjustments from being misclassified as clicks. With this design, users can operate the GUI in a manner analogous to a physical touchpad—press to move, lift to stop, and tap to click—with a very low learning cost.

During clicking, the cursor should ideally remain stationary. However, our tests showed that between the "Press" and "Lift" events of a click, the values predicted by TraceRing are small and only move the cursor by a few pixels. For convenience, we did not filter out this jitter. Users may notice a very slight cursor movement when clicking, but its impact on click accuracy is negligible. We consider this an engineering issue that should be addressed in future work to further improve the user experience.

Figure 7: In the user encoder, we use a weighted sum to assess the importance of different time steps for distinguishing users. Here, we visualize it, where darker colors indicate that the corresponding data segment is more important for differentiating users.

5.4.2 Real-time Inference. We set a default user embedding to support cold-start usage. During user interaction, once 3000 frames of data (15 seconds) are accumulated, we trigger a user embedding update. This computation runs in a separate process to avoid interfering with the main process's runtime performance. The resulting embedding is then updated using an exponential moving average (EMA), i.e.:

$$emb_t = (1-\alpha)\, emb_{t-1} + \alpha\, emb_t^{\text{new}}, \quad \alpha \in (0,1) \tag{8}$$

Here, emb_t denotes the user embedding at time t. For the first update, we set $\alpha = 0.8$; for subsequent updates, we use $\alpha = 0.2$. This ensures that the embedding closely reflects the user's accurate representation while remaining robust against extreme fluctuations.

We evaluated the runtime efficiency of the complete algorithm on a laptop equipped with an Intel(R) Core(TM) i7-11800H CPU (without GPU). Without any optimization, the system runs at 200 fps on CPU alone. We believe the approach is capable of real-time performance and, with additional optimizations such as quantization, pruning, and platform-specific acceleration, can achieve full-speed operation on virtually any platform.

5.4.3 Filters and CD Gain. The model outputs velocities on the plane, which can be considered the raw touchpad input. To convert these into actual cursor movements, a CD gain needs to be applied. Additionally, we employ a 1-Euro filter [4] to smooth the velocities, a technique widely used in pointing tasks. After tuning, we set the 1-Euro filter's minimum cutoff frequency to 0.0004 and the speed coefficient to 0.08. The CD gain is configured as follows:

$$CD(v_t) = CD_{\min} + \left(CD_{\max} - CD_{\min}\right) \cdot \frac{1}{1 + e^{-k_v \cdot (v_t - v_0)}} \tag{9}$$

Here, v_t denotes the magnitude of the velocity at the current timestep. The parameters are set as follows: $CD_{\min} = 0.3$, $CD_{\max} = 2.0$, $k_v = 0.2$, and $v_0 = 15$.

6 Usability Evaluation

Figure 8: We built a Fitts' Law usability testing system and used a touchpad and an airmouse as baselines. (a) The visualization interface and setup used in the experiment. (b) The airmouse used in the study. (c) The mouse used in the study.

We conducted a study to evaluate the pointing performance of TraceRing by comparing it with conventional pointing devices (Mouse and Touchpad) as well as an IMU-based pointing device (AirMouse). In addition, we examined the performance of TraceRing across different surface types.

6.1 Apparatus

In this experiment, we used the same smart ring and wearing configuration as in the previous study (Sec. 4.1). The entire system was deployed on a Lenovo Legion Y9000P 2021 equipped with an Intel(R) Core(TM) i7-11800H CPU. The GPU was disabled, and the model was fully executed on the CPU. For comparison, we included three baselines: the built-in touchpad of the laptop, a wireless mouse (Xiaomi XMSMSB01YM) and an air mouse that also uses IMU-based cursor control (Deli 2803). The experimental setup is shown in Figure 8.

6.2 Participants

We recruited 16 participants (10 males, 6 females, aged 22–36, M=26.56) from a university campus and its surrounding community. All participants were right-handed; 13 were highly familiar with touchpad use, and 6 had prior experience with the air mouse. None of them overlapped with the participants from the previous data collection experiment.

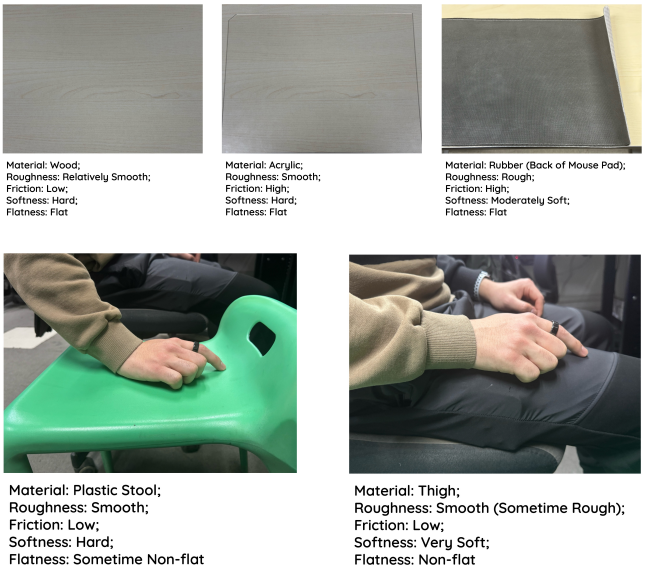


Figure 9: The experimental setup with different surfaces. The Fitts' law experiment was conducted on the wood surface, while the remaining surfaces were provided for participants to try and to give subjective ratings.

6.3 Design and Procedure

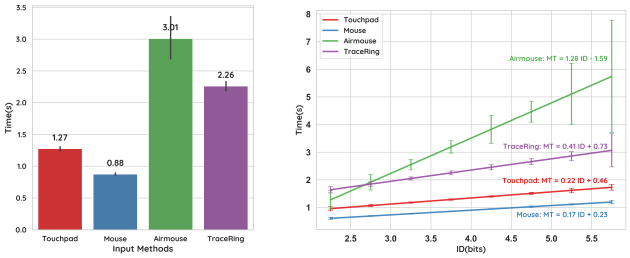
To evaluate the actual effectiveness of cursor control, we conducted a Fitts' Law experiment comparing three input methods: the laptop touchpad, the air mouse, and TraceRing. We developed a Python-based experimental interface (Fig. 8(a)). In each trial, two yellow circular buttons appeared on the screen, one labeled S (Start) and the other E (End). The button diameter ranged from 45 px to 120 px, and the distance between the two buttons ranged from 45 px to 1500 px (screen resolution: 1920 × 1080 px). Participants were required to first move the cursor to S and click the button , after which the button turned green. They then had to move as quickly as possible to E. A click at E was considered a success, after which the next trial began. Besides task-related information (e.g., target size, distance, and timestamps), we logged the full cursor trajectories to enable later reconstruction of the experimental process.

Before the experiment started, participants were given about 5 minutes to familiarize themselves with all four input methods. They then completed the four conditions in a Latin-square counterbalanced order . They repeated 5(rounds) × 10(trials)=50(trials) for each condition. After each round, participants rested for about 30 seconds.

After the target selection experiment, we additionally asked participants to try TraceRing on surfaces with different materials. All materials are listed in Fig. 9. For each material, participants were required to use TraceRing on the surface for at least 1 minute, including moving the cursor and clicking, until they felt able to evaluate its performance under that condition.

The entire experiment lasted about 45 minutes. After each round, participants filled out a subjective questionnaire and answered several open-ended questions. Each participant received a 15 dollar reward for their participation.

6.4 Results



(a) The mean completion time of different input methods.

(b) The linear relationship between ID and the completion time.

Figure 10: Visualizations of the average completion time across systems, as well as the completion time of each system under different IDs. Vertical error bars represent the 95% standard error of the mean (SEM).

6.4.1 Task Completion Efficiency. Figure 10(a) compares the mean completion times across different systems. It can be observed that the mean completion times for TraceRing (MT = 2.26s, SD = 0.91s) are substantially shorter than for the Airmouse (MT = 3.01s, SD = 4.16s), but longer than Touchpad (MT = 1.27s, SD = 0.38s) and Mouse (MT = 0.88s, SD = 0.24s). A one-way ANOVA further confirmed these results: the completion time of the TraceRing was significantly shorter than that of the Airmouse ($F=17.85$, $p<0.001$), but significantly longer than that of the Touchpad ($F=574.74$, $p<0.001$) and the Mouse ($F=1217.30$, $p<0.001$).

Since task difficulty has a significant effect on completion time, direct comparisons of raw times are not fair. Following Fitts' Law, we quantify task difficulty using the index of difficulty (ID), defined

as $ID = \log_2 \frac{2D}{W}$, where D is the target distance and W is the target width. According to Fitts' Law, completion time is expected to exhibit a linear relationship with ID. Figure 10(b) presents the linear regression of completion times across different difficulty levels. The results show that TraceRing performs comparably to AirMouse when the ID is below 3, but substantially outperforms AirMouse at higher IDs. This is also reflected in the slope of the fitted line, indicating that the input efficiency of TraceRing is considerably better than that of AirMouse. Although its input efficiency does not yet reach the level of the touchpad or mouse, we still consider TraceRing's performance quite promising. Compared to the single-ring configuration of MouseRing[47], whose input efficiency is significantly lower than that of AirMouse, TraceRing represents a substantial improvement.

In addition, relative to the experimental setup of MouseRing[47], we replaced dwell-based selection with click-based selection, making the pointing evaluation more closely resemble real-world use. However, since click is not a core contribution of this work, we only provide a simple implementation in Section 5.4.1. This implementation works well for most users, but in our study we observed that a subset of participants exhibited relatively low click recognition accuracy (about 60%–70%). These users tended to wait for feedback after clicking before deciding whether to continue moving or click again, which increased their cognitive load and reaction time. For the other devices, click accuracy was close to 100%, so this issue did not occur. Improving click robustness will therefore be a key focus of future optimization to make TraceRing a sufficiently practical pointing device.

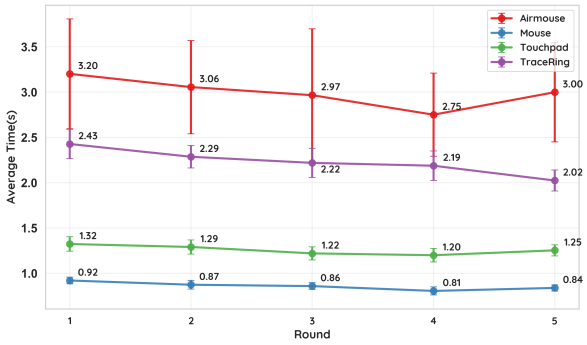


Figure 11: Completion time of each system under different rounds.

6.4.2 Learning Efficiency. To illustrate the learning efficiency of different input methods, we plotted the completion times across five rounds (Figure 11). As shown, all four input methods exhibited a slight decrease in completion time with increasing rounds, but the reduction was not substantial. This indicates that TraceRing is easy to learn and requires little training effort, but still needs some time to adapt to achieve better performance.

6.4.3 Performance Across Users. Figure 12 plots the mean completion time for each participant using TraceRing. As shown, Participants 3, 5, and 8 exhibit noticeably higher mean times than

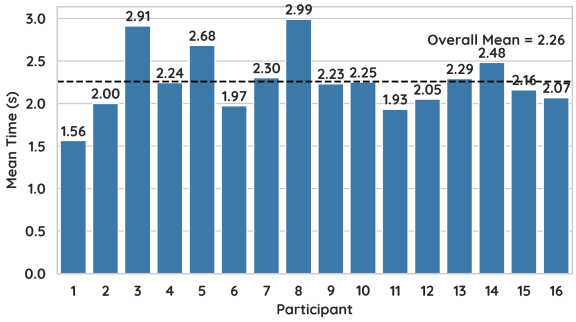


Figure 12: Completion time of TraceRing across users.

the overall average. Participants 3 and 8 reported relatively low click recognition accuracy (around 70%–80%), which forced them to frequently wait for feedback after clicking and then click again, making their response times difficult to estimate. Participant 5 showed slightly lower pointing accuracy than the others, which may be attributable to individual factors (because his fingers are relatively thick, the ring can only be worn near the proximal interphalangeal (PIP) joint rather than at the base of the finger); for example, there may be relatively few users in the training data whose behavior and physiology are similar to theirs. Nevertheless, because most participants' completion times are relatively consistent, we believe that TraceRing can effectively accommodate diverse individual characteristics and provide a robust pointing input experience. For participants with poorer performance, we expect that augmenting the dataset could help cover a broader range of cases.

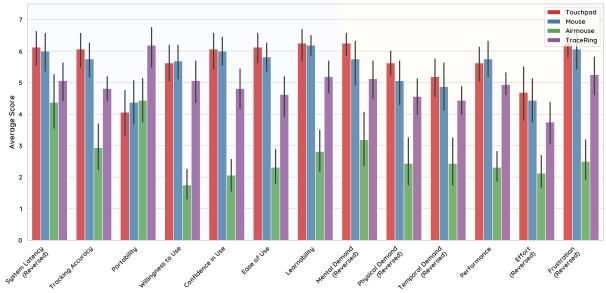


Figure 13: Subjective scale results of different input methods in the cursor control experiment. Some items were reverse-scored so that, for each item, a higher score indicates better system performance.

6.4.4 Subjective Indicators. We selected and adapted several items from the SUS (System Usability Scale) (left half) and combined them with all items from the NASA-TLX scale (right half) to form a comprehensive scale for evaluating subjective scores across different systems (Figure 13). As shown, although TraceRing does not fully match the Touchpad in terms of latency or tracking accuracy, its higher portability and ease of learning lead to great user willingness to use it. The NASA-TLX results further indicate that TraceRing imposes a significantly lower overall workload than the Airmouse(

mean score 4.68 v.s. 2.50, F=130.60, p<0.001), but higher than the Touchpad and Mouse, with absolute workload values remaining relatively low.

We also collected participants' willingness to use TraceRing and asked them to suggest potential usage scenarios. Among the participants, 87.5% (14/16) expressed that they would be willing to use TraceRing immediately in daily life. They envisioned using it in scenarios such as mobile work/iPad (6/14), large screens/presentations (5/14), remote control (5/14), narrow desktop (2/14), multi-screen interaction (1/14), privacy input (1/14) and control device on the leg (1/14). Representative comments include: "It feels very good overall to me, because using it is very similar to using a touchpad." (P2); "I usually use a mouse, so I was not very used to TraceRing at first. After adapting, I found that its accuracy in my hands is no worse than most mice." (P5); "When I am lying in a chair or on the bed watching a movie, I would really like to use it, so that I do not have to get up to pause and play." (P6); "The movement amplitude is sometimes different from what I expect, so it may take some getting used to. Its large-range control is similar to a touchpad, but there is still room for improvement in fine-grained adjustments over small ranges." (P10); "It is very convenient to carry and very easy to learn, so it may be particularly suitable for middle-aged and older users." (P16)

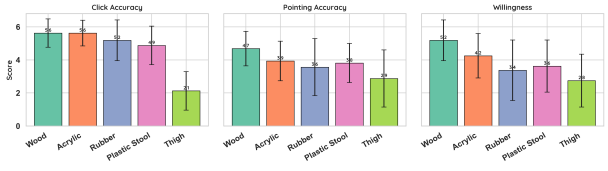


Figure 14: Subjective scale results of different types of surfaces.

6.4.5 Performance on Different Types of Surfaces. As shown in Fig. 14, we asked participants to evaluate click accuracy, pointing accuracy, and willingness to use TraceRing on different surfaces. On relatively hard surfaces, click accuracy did not change noticeably, whereas on a very soft surface (the thigh), click accuracy dropped substantially. For pointing accuracy, rougher surfaces with higher friction had some impact, but the effect was not particularly large. However, because the click response rate on the thigh was low, many participants found it difficult to initiate cursor control via pressing, which in turn led to a large decrease in movement accuracy. In contrast, on a plastic stool at approximately the same height as the thigh, click and pointing accuracy were much higher, even though the surface was not flat. Given that our dataset only covers movement and tapping on hard, smooth surfaces, these results are reasonable. Tap detection relies on the contact impulse, which differs across surfaces: hard surfaces produce sharp IMU spikes, while soft surfaces produce more sustained oscillations, limiting cross-surface generalization. Surface differences are less pronounced for velocity estimation during continuous motion. Collecting data on a wider range of surfaces could improve usability, which we leave for future work.

7 Application

This section illustrates typical usage scenarios of TraceRing. Because its interaction model mirrors that of a conventional touchpad, its applications are very similar. However, as a smart ring that can be worn continuously, TraceRing is inherently portable and always available, offering additional convenience in certain contexts. These examples are not proposed as novel interaction techniques; rather, they are intended to help readers better understand the potential application space of TraceRing.

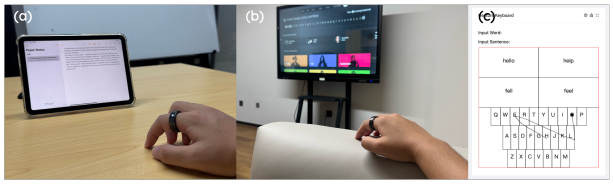


Figure 15: TraceRing's diverse application scenarios. (a) Mobile productivity with a tablet. (b) Remote control of a large display. (c) Text input using gesture typing.

7.1 Mobile and Ubiquitous Computing

New interaction bottlenecks have emerged with the growing prevalence of devices like tablets, foldable smartphone and AR/VR headsets. Direct touch interaction becomes difficult when a tablet or a foldable smartphone is propped up on a stand to serve as a display. Meanwhile, in AR/VR environments, camera-based hand tracking is not only power-intensive, but its corresponding mid-air interactions also lead to fatigue and inconsistent precision due to the lack of physical feedback. TraceRing offers a unified and elegant solution for these scenarios by transforming any regular physical surface into an on-demand, high-precision trackpad. For instance, users can perform precise actions on a propped-up tablet, such as play/pause or scrolling, by simply sliding their finger on the desk.

7.2 Remote Control of Large Displays and IoT Devices

In the past, interaction with large distant displays or IoT devices has typically relied on remote controllers, which are often misplaced, and in many situations users may also prefer not to interact via direct touch at close range. TraceRing provides an always-available solution for remote interaction: users can perform convenient and precise control on a nearby surface while remaining in a comfortable posture on a bed or sofa. By connecting via Bluetooth to a smartphone or control hub, TraceRing can easily interact with any target device.

7.3 Gesture and Text Input

Pointing alone is insufficient for interactions with large displays and AR environments; gesture and text input are often also required. Since TraceRing provides precise cursor control, it can naturally be extended to support gesture and text input. By leveraging existing 2D gesture recognizers [52], continuous cursor movements can be directly converted into gestures. Similarly, this enables gesture typing for fast text entry. Following the approach in [60], each

input sequence starts from the letter "G," and the user moves sequentially to the desired letters, lifting the finger to confirm each selection. This method compensates for TraceRing's limitation of only providing relative displacement.

Based on these designs, we provide users with a complete and universal input solution: a daily-wearable ring that can handle pointing, gesture, and text input on large screens or other devices without additional controllers, offering a more convenient interaction experience in everyday life.

8 Discussion

8.1 Personalization–Generalization Trade-off

Generalization is a critical goal in all machine learning tasks, especially for real-world applications. In domains such as NLP and CV, it has been well established that sufficient data and computational resources can lead to strong generalization. However, in deep learning with sensor data, training data is often scarce making achieving robust generalization challenging: the model's predictions may easily fail when switching to a new user, a new environment, or even when the wearing orientation of the device slightly changes.

On the other hand, sensor-based interaction has its unique advantage: the same device typically belongs to a single user, meaning that high universality and generalization are not always necessary. This creates room for personalization. This paper explores the necessity of personalization in achieving touchpad-like pointing with a single IMU ring, and propose an online calibration method based on a User Encoder and MoE architecture, achieving personalization with minimal additional computational cost. We hope our approach can inspire future work on personalization for deep learning tasks in sensor-based interaction.

8.2 Closing the Gap to Touchpad Input

TraceRing still falls short of delivering an actual touchpad-like experience from subjective feedback and objective performance. When asked about the reasons, participants highlighted two main issues. The first is limited control precision, which is particularly noticeable when selecting smaller targets. Although the cursor generally follows the intended direction, overshooting occasionally occurs. Because TraceRing relies entirely on deep learning to predict velocity, the same action may produce different outcomes, and small variations in motion can lead to unintentional deviations. Also, this issue arises partly because the model output lags behind the actual finger motion, and partly due to delays in the system, causing the cursor to feel less responsive and harder to stop precisely. The second issue is that some users found the default cursor speed (Dots Per Inch, DPI) either too fast or too slow, expressing a preference for adaptive adjustment.

To address the first issue, several rule-based system optimizations could be introduced to improve control performance. For example, detecting abrupt stopping behaviors through rules (rather than relying solely on the model) could help correct the output and reduce latency between user intention and system response, thereby improving controllability. Furthermore, our current system transmits data via BLE at 20 packets per second, with each packet containing 10 frames. While sufficient, this sampling rate may still be low for precise cursor control. Meanwhile, on an i7-11800H, a single model forward pass in PyTorch takes 0.0014 s, and one call to pynput.mouse.Controller().move() takes 0.0012 s. Consequently, when a data packet is received, there is at least a 76 ms gap between the time its last frame is actually sampled and the time it is processed and the cursor is moved, which may contribute to interaction latency. Improvements in communication protocols could further reduce latency and enhance the user experience.

To address the second issue, multiple DPI levels should be introduced for user selection. Another promising approach is to adopt an online calibration method (e.g., [30]), which adaptively tunes the CD gain in real time to provide each user with a cursor control speed better suited to their preferences.

8.3 Practical Deployment

We first analyzed the system in terms of parameter count and computational cost. The entire model contains approximately 340K parameters, with the Task Encoder accounting for 150K, a single MLP Expert 4K, and the User Encoder 160K. This model size is comparable to that of prior edge-computing work[66]. Assuming an input rate of 200 Hz, the Task Encoder and MLP Experts run 200 times per second, while the User Encoder runs once every 15 seconds. This corresponds to computational loads of 19 MFLOPs for the Task Encoder, 0.85 MFLOPs per MLP Expert, and 0.05 MFLOPs for the User Encoder. Overall, the parameter count and computation are extremely low, and the personalized components (MLP Experts and User Encoder) consume exceptionally minimal resources. This means that the model can be possibly deployed on most edge devices, such as smartphones or tablets, even the ring itself.

9 Conclusion

We presented TraceRing, a single-IMU ring that enables touchpad-like pointing through personalized learning. By analyzing a large-scale dataset with precise annotations, we identified the fundamental limitation of prior single-IMU approaches and proposed a multi-task personalization framework that adapts to users without explicit calibration or fine-tuning. Our evaluation shows that TraceRing achieves 33.9% performance gains over state-of-the-art baseline and approaches much better efficiency to airmouse in real-time tasks (2.26s v.s. 3.01s in average task completion time), while participants valued its portability and ease of use. Moreover, TraceRing naturally extends to gesture and text input, offering a complete and universal input solution for large displays, AR/VR, and mobile computing. This work not only demonstrates the feasibility of practical IMU-based pointing but also highlights the potential of lightweight personalization in sensor-based machine learning, paving the way toward ubiquitous and efficient wearable interaction.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2024YFB4505500 & 2024YFB4505501, the National Natural Science Foundation of China under Grant No. 62502263.

References

[1] Roger Boldu, Alexandru Dancu, Denys JC Matthies, Pablo Gallego Cascón, Shanaka Ransir, and Suranga Nanayakkara. 2018. Thumb-in-motion: Evaluating thumb-to-ring microgestures for athletic activity. In *Proceedings of the 2018 ACM Symposium on Spatial User Interaction*. 150–157.

[2] Yifeng Cao, Ashutosh Dhekne, and Mostafa Ammar. 2023. ViSig: Automatic interpretation of visual body signals using on-body sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 1 (2023), 1–27.

[3] Géry Casiez and Nicolas Roussel. 2011. No more bricolage! Methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 603–614.

[4] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2527–2530. doi:10.1145/2207676.2208639

[5] Yidong Chai, Haoxin Liu, Hongyi Zhu, Yue Pan, Anqi Zhou, Hongyan Liu, Jianwei Liu, and Yang Qian. 2024. A profile similarity-based personalized federated learning method for wearable sensor-based human activity recognition. *Information & Management* 61, 7 (2024), 103922.

[6] Ishan Chatterjee, Jiexin Ding, Anandghan Waghmare, Joseph Breda, Yuquan Deng, Bo Liu, Yuntao Wang, and Shwetak Patel. 2025. FlowRing: Integrated Microgesture and Surface Interaction Ring for Versatile XR Input. *Proc. ACM Hum.-Comput. Interact.* 9, 5, Article MHCI010 (Sept. 2025), 28 pages. doi:10.1145/3743706

[7] Ke-Yu Chen, Shwetak N Patel, and Sean Keller. 2016. Finexus: Tracking precise motions of multiple fingertips using magnetic sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1504–1514.

[8] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. 2021. An adaptive model of gaze-based selection. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–11.

[9] Carlo Colombo, Alberto Del Bimbo, and Alessandro Valli. 2003. Visual capture and understanding of hand pointing actions in a 3-D environment. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 33, 4 (2003), 677–686.

[10] Guillem Cornella-Barba, Andria J Farrens, Christopher A Johnson, Luis Garcia-Fernandez, Vicky Chan, and David J Reinkensmeyer. 2024. Using a Webcam to Assess Upper Extremity Proprioception: Experimental Validation and Application to Persons Post Stroke. *Sensors* 24, 23 (2024), 7434.

[11] Osamu Fukuda, Jun Arita, and Toshio Tsuji. 2006. An EMG-controlled omnidirectional pointing device. *Systems and Computers in Japan* 37, 4 (2006), 55–63.

[12] Patrick Grady, Chengcheng Tang, Samarth Brahmbhatt, Christopher D. Twigg, Chengde Wan, James Hays, and Charles C. Kemp. 2022. PressureVision: Estimating Hand Pressure from a Single RGB Image. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 328–345. doi:10.1007/978-3-031-20068-7_19

[13] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and low-latency sensing of touch contact on any surface with finger-worn IMU sensor. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 1059–1070.

[14] Yunjo Han, Panyu Zhang, Minseo Park, and Uichin Lee. 2024. Systematic evaluation of personalized deep learning models for affect recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 4 (2024), 1–35.

[15] Youfang Han, Wei Zhao, Ge Gao, Xiangjin Chen, Jiliang Yin, Lin Wang, Xin Meng, Yang Yu, and Tengxiang Zhang. 2025. DCSNN: An Efficient and High-speed sEMG-based Transient-state Micro-gesture Recognition Method on Wearable Devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 9, 2 (2025), 1–34.

[16] Shogo Hanayama, Riku Kitamura, Takumi Yamamoto, Takashi Amesaka, Liwei Chan, and Yuta Sugiura. 2025. SkinRing: Ring-shaped Device Enabling Wear Direction-Independent Gesture Input on Side of Finger. In *2025 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 386–392.

[17] Zhe He, Zixuan Wang, Chun Yu, Chengwen Zhang, Xiyuan Shen, and Yuanchun Shi. 2025. WritingRing: Enabling Natural Handwriting Input with a Single IMU Ring. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–15.

[18] Anuradha Herath, Bradley Rey, Sandra Bardot, Sawyer Rempel, Lucas Audette, Huizhe Zheng, Jun Li, Kevin Fan, Da-Yuan Huang, Wei Li, et al. 2022. Expanding Touch Interaction Capabilities for Smart-rings: An Exploration of Continual Slide and Microroll Gestures. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–7.

[19] Chenghao Huang, Xiaolu Chen, Yanru Zhang, and Hao Wang. 2024. Fedcrl: Personalized federated learning with contrastive shared representations for label heterogeneity in non-iid data. *arXiv preprint arXiv:2404.17916* (2024).

[20] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[21] Tetsuya Itou, Muneaki Terao, Junji Nagata, and Masaki Yoshida. 2001. Mouse cursor control system using EMG. In *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vol. 2. IEEE, 1368–1369.

[22] Patrick Kaifosh, Thomas R. Reardon, Brian D. Allen, Chris Anderson, Sacha Arnoud, Rahul Arora, Mridu Atray, Lana Awad, Francisco Ayerbe, Christopher Baker, Nicholas Baker, Alexandre Barachant, Philip Bard, Wilman Pimentel Beltran, Adam Berenzweig, Rohin Bhasin, Joe Bienkowski, Sean Bittner, Luke Boegner, Anu Bolarinwa, Don Bosley, Matthew Bracaglia, Mario Bräcklein, Maclyn Brandwein, Joe Bravate, Matt Butler, Adam J. Calhoun, Chia-Jung Chang, Daniel Chenet, Joshua Chester, Rudi Chiarito, Rohan Chitnis, John Choi, Won Chun, Jeremiah Chung, James Connors, Jota Costa, Mark Cramer, Raven Cunningham, William F. Cusack, Nathan Danielson, Thomas J. Davidson, Bruno De Araujo, Bob DiMaiolo, Scott Draves, Alan Du, Zaina Edelson, Phina Enemuo, Mina Fahmi, Nariman Farsad, Ali Farshchian, Randy Feliz, Jake Fine, Emanuele Formento, Dustin Freeman, Jianing Fu, Jean-Christophe Gagnon-Audet, Rupesh Gajurel, Jonathan Gamutan, Sida Gao, Jonateal Garcia, Nathalie Therese Helene Gayraud, Minha Ghani, Sayan Ghosh, Vickram Gidwani, Danny Giebisch, Greg Gimler, Alexandre Gramfort, Lauren Grosberg, Bryn Gunther, Ning Guo, Chetan Gupta, Sinem Guven Kaya, Austin Ha, Katarina Hadjer, Carlos Xavier Hernández, Stav Hertz, Carl Hewitt, Daniel N. Hill, Kirak Hong, Lillian Hong, Helen Hou, Stepan Hruda, Alex Hsieh, Vivian Hsiung, Rongqing Huang, Yue Hui, Hazel Hulet, Shaker Islam, Vinay Jayaram, Connie Jiang, Xiaodong Jiang, Brooke Juarez, James Jaeyoon Jun, Na Young Jun, Nirag Kadakia, Nishant Kakar, Ajay Kamdar, Ta-Chu Kao, Steven Kober, TW Koh, Christina Shabu Koshy, Andrzej Lawn, Claire Lee, Jennifer Lee, JinHyung Lee, Juheui Amy Lee, Tiffanie Li, Jonathan Liao, Yingru Liu, Yuxuan Liu, Saar Lively, Kati London, Roddy Louie, Francisco Luongo, Attila Maczak, Niru Maheswaranathan, Michael Mandel, Jesse Marshall, Najja Marshall, Mirek Martincik, Nicolas Yvan Masse, Stephen McAnearney, Ashley McHugh, Jorge Aurelio Menendez, Josh Merel, David Miller, Ilya Milyavskiy, Ricardo Pio Monti, Sean Moore, Yonathan Morin, Brock Morrell, Dano Morrison, Anthony Moschella, Suman Mulumudi, Conner Muth, Krunal Naik, Norris Nakagaki, Ajay Nathan, Romario Nelson, Jimson Ngeo, Keven Nguyen, Luke O'Connor, Shay Ohayon, Garrick Orchard, Chris Osborn, Timothy M. Otchy, Emmanuella Owolabi, Adam M. Packer, Tejaswy Pailla, Julia Paredes, Sean Parker, Diogo Peixoto, Matias Perez, Zavion Perez, Adrien Piérard, Stephen M. Plaza, Natalie Plotkin, Eftychios Pnevmatikakis, Brandon Pool, Shanil Puri, Sunaina Rajani, Jose Ramirez Fuentes, Julian Ramos Rojas, Tanvi Ranjan, Devin Reardon, Jonathan Reid, Jason Reisman, Lain Warawao Nemo Mora y Rivera, Sebi Rolotti, Andrew Rosenkranz, Ian Roth, Likhon Roy, Ran Rubin, Alexander Rudnicki, Sam Russell, Abby Russo, James Sacra, Amir Sadoughi, Roxanna Salim, Aichatou Savane, Collin Schlager, David Schwab, Jeffrey Seely, Mike Seltzer, Nurettin Dorukhan Sergin, Ami Shah, Anish Shah, Philip Shamash, Vandita Sharma, Stephie Shen, Kevin Shi, Olivia Shiah, Yasmin Siahpoosh, Noor Siddiqi, Jeremy Simpson, Gagandip Singh, Viswanath Sivakumar, Jeff Smith, Seyyid Emre Sofuoglu, Ivy Jiyoung Song, Morgan Springer, Adrian Spurr, Fabio Stefanini, Connor Stout, Emanuel Strauss, Swetha Suresh, Ananya Suri, David Sussillo, Ziyi Tang, Vikram Tank, Jesslyn Tannady, Aliqyan Tapia, Tugce Tasci, Tiberiu Tesileanu, Aman Tiwari, Anoushka Tiwari, Calvin Tong, Blizelle Tormis, Julia Trabulsi, Migmar Tsering, Kyle Urquhart, Peter Walkington, Megan Wang, Renxiong Wang, Zhuo Wang, Christy Warden, Richard Warren, Claire L. Warriner, Ron J. Weiss, Daniel Z. Wetmore, Ezri White, Christopher Wiebe, Steve Williams, Yuguan Xing, Chris Ye, Akshay Yembarwar, Shuibenyang Yuan, Michael Zawadzki, Mingrui Zhang, Jiesi Zhao, Kevin Zheng, Joseph Zhong, Lei Zhou, Danny Zlobinsky, and CTRL-labs at Reality Labs. 2025. A generic non-invasive neuromotor interface for human-computer interaction. *Nature* 645, 8081 (Sept. 2025), 702–711. doi:10.1038/s41586-025-09255-w

[23] Farid Karimli, Hao Yu, Srishti Jain, Emmanuel Sarpong Akosah, Margrit Betke, and Wenxin Feng. 2024. Demonstration of CameraMouseAI: A Head-Based Mouse-Control System for People with Severe Motor Disabilities. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility*. 1–6.

[24] Keiko Katsuragawa, Krzysztof Pietroszek, James R Wallace, and Edward Lank. 2016. Watchpoint: Freehand pointing with a smartwatch in a ubiquitous display environment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*. 128–135.

[25] Bulat Khaertdinov and Stylianos Asteriadis. 2023. Explaining, analyzing, and probing representations of self-supervised learning models for sensor-based human activity recognition. In *2023 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 1–10.

[26] Wolf Kienzle and Ken Hinckley. 2014. LightRing: always-available 2D input on any surface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 157–160.

[27] Joowan Kim, Minkyu Kim, and Keehoon Kim. 2016. Development of a wearable HCI controller through sEMG & IMU sensor fusion. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 83–87.

[28] Kazuaki Kondo, Genki Mizuno, and Yuichi Nakamura. 2016. Analysis of Human Pointing Behavior in Vision-based Pointing Interface System-difference of two typical pointing styles. *IFAC-PapersOnLine* 49, 19 (2016), 367–372.

[29] Daniel Laidig and Thomas Seel. 2023. VQF: Highly accurate IMU orientation estimation with bias estimation and magnetic disturbance rejection. *Information Fusion* 91 (2023), 187–204. doi:10.1016/j.inffus.2022.10.014

[30] Byungjoo Lee, Mathieu Nancel, Sunjun Kim, and Antti Oulasvirta. 2020. Auto-Gain: 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. doi:10.1145/3313831.3376244

[31] Royson Lee, Minyoung Kim, Da Li, Xinchi Qiu, Timothy Hospedales, Ferenc Huszár, and Nicholas Lane. 2023. Fedl2p: Federated learning to personalize. *Advances in Neural Information Processing Systems* 36 (2023), 14818–14836.

[32] Chenglin Li, Di Niu, Bei Jiang, Xiao Zuo, and Jianming Yang. 2021. Meta-har: Federated representation learning for human activity recognition. In *Proceedings of the web conference 2021*. 912–922.

[33] Yifan Li, Masaaki Fukumoto, Mohamed Kari, Tomoyuki Yokota, Takao Someya, Yoshihiro Kawahara, and Ryo Takahashi. 2025. Demo of picoRing mouse: ultra-low-powered wireless mouse ring with ring-to-wristband coil-based impedance sensing. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–5.

[34] Youpeng Li, Xuyu Wang, and Lingling An. 2023. Hierarchical clustering-based personalized federated learning for robust and fair human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 1 (2023), 1–38.

[35] Yang Liu, Chengdong Lin, and Zhenjiang Li. 2021. WR-Hand: Wearable armband can track user's hand. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27.

[36] I. Scott MacKenzie and Shaidah Jusoh. 2001. An Evaluation of Two Input Devices for Remote Pointing. In *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction (EHCI '01)*. Springer-Verlag, Berlin, Heidelberg, 235–250.

[37] Abdullah Mueen, Sheng Zhong, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. 2022. The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[38] Mathieu Nancel, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Pourang P. Irani, and Michel Beaudouin-Lafon. 2013. High-precision pointing on large wall displays using small handheld devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 831–840. doi:10.1145/2470654.2470773

[39] Kai Nickel and Rainer Stiefelhagen. 2003. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *Proceedings of the 5th international conference on Multimodal interfaces*. 140–146.

[40] Ignacio Oguiza. 2023. tsai - A state-of-the-art deep learning library for time series and sequential data. Github. https://github.com/timeseriesAI/tsai

[41] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2021. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042* (2021).

[42] Ju Young Oh, Jun Lee, Joong Ho Lee, and Ji Hyung Park. 2017. Anywheretouch: Finger tracking method on arbitrary surface using nailed-mounted imu for mobile hmd. In *International Conference on Human-Computer Interaction*. Springer, 185–191.

[43] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Guoliang Xing, and Jianwei Huang. 2022. Clusterfl: A clustering-based federated learning system for human activity recognition. *ACM Transactions on Sensor Networks* 19, 1 (2022), 1–32.

[44] Farshid Salemi Parizi, Eric Whitmire, and Shwetak Patel. 2019. Auraring: Precise electromagnetic finger tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–28.

[45] Riccardo Presotto, Gabriele Civitarese, and Claudio Bettini. 2022. Semi-supervised and personalized federated activity recognition based on active learning and label propagation. *Personal and Ubiquitous Computing* 26, 5 (2022), 1281–1298.

[46] Javier San Agustin, John Paulin Hansen, Dan Witzner Hansen, and Henrik Skovsgaard. 2009. Low-cost gaze pointing and EMG clicking. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. 3247–3252.

[47] Xiyuan Shen, Chun Yu, Xutong Wang, Chen Liang, Haozhan Chen, and Yuanchun Shi. 2024. MouseRing: Always-available Touchpad Interaction with IMU Rings. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–19.

[48] Taoran Sheng and Manfred Huber. 2020. Weakly supervised multi-task representation learning for human activity analysis using wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–18.

[49] Jiankai Tang, Zhe He, Mingyu Zhang, Wei Geng, Chengchi Zhou, Weinan Shi, Yuanchun Shi, and Yuntao Wang. 2026. ?-Ring: A Smart Ring Platform for Multimodal Physiological and Behavioral Sensing. In *Companion of the 2025 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Finland) *(UbiComp Companion '25)*. Association for Computing Machinery, New York, NY, USA, 1271–1277. doi:10.1145/3714394.3756252

[50] Hsin-Ruey Tsai, Min-Chieh Hsiu, Jui-Chun Hsiao, Lee-Ting Huang, Mike Chen, and Yi-Ping Hung. 2016. TouchRing: subtle and always-available input using a multi-touch ring. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 891–898.

[51] Toshio Tsuji, Osamu FUKUDA, Mitsuru MURAKAMI, and Makoto KANEKO. 2001. An EMG controlled pointing device using a neural network. *Transactions of the Society of Instrument and Control Engineers* 37, 5 (2001), 425–431.

[52] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2018. $Q: a super-quick, articulation-invariant stroke-gesture recognizer for low-resource devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Barcelona, Spain) *(MobileHCI '18)*. Association for Computing Machinery, New York, NY, USA, Article 23, 12 pages. doi:10.1145/3229434.3229465

[53] Daniel Vogel and Ravin Balakrishnan. 2005. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 33–42.

[54] Haopeng Wang, Ludwig Sidenmark, Florian Weidner, Joshua Newn, and Hans Gellersen. 2025. HeadShift: Head Pointing with Dynamic Control-Display Gain. *ACM Trans. Comput.-Hum. Interact.* 32, 1, Article 2 (April 2025), 28 pages. doi:10.1145/3689434

[55] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern recognition letters* 119 (2019), 3–11.

[56] Andrew D Wilson and Edward Cutrell. 2005. Flowmouse: A computer vision-based pointing and gesture input device. In *IFIP Conference on Human-Computer Interaction*. Springer, 565–578.

[57] Yingjing Xiao, ZhiChao Huang, and Yang Gao. 2025. From Wrist to Finger: Hand Pose Tracking Using Ring-Watch Wearables. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–7.

[58] Vasco Xu, Chenfeng Gao, Henry Hoffmann, and Karan Ahuja. 2024. Mobileposer: Real-time full-body pose estimation and 3d human translation from imus in mobile consumer devices. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–11.

[59] Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2012. Magic finger: always-available input through finger instrumentation. 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, 147–156. doi:10.1145/2380116.2380137

[60] Zhican Yang, Chun Yu, Xin Yi, and Yuanchun Shi. 2019. Investigating Gesture Typing for Indirect Touch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article 117 (Sept. 2019), 22 pages. doi:10.1145/3351275

[61] Hui-Shyong Yeo, Juyoung Lee, Hyung-il Kim, Aakar Gupta, Andrea Bianchi, Daniel Vogel, Hideki Koike, Woontack Woo, and Aaron Quigley. 2019. Wrist: Watch-ring interaction and sensing technique for wrist gestures and macro-micro pointing. In *Proceedings of the 21st international conference on human-computer interaction with mobile devices and services*. 1–15.

[62] Tianhong Catherine Yu, Guilin Hu, Ruidong Zhang, Hyunchul Lim, Saif Mahmud, Chi-Jung Lee, Ke Li, Devansh Agarwal, Shuyang Nie, Jinseok Oh, et al. 2024. Ring-a-Pose: A Ring for Continuous Hand Pose Tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 4 (2024), 1–30.

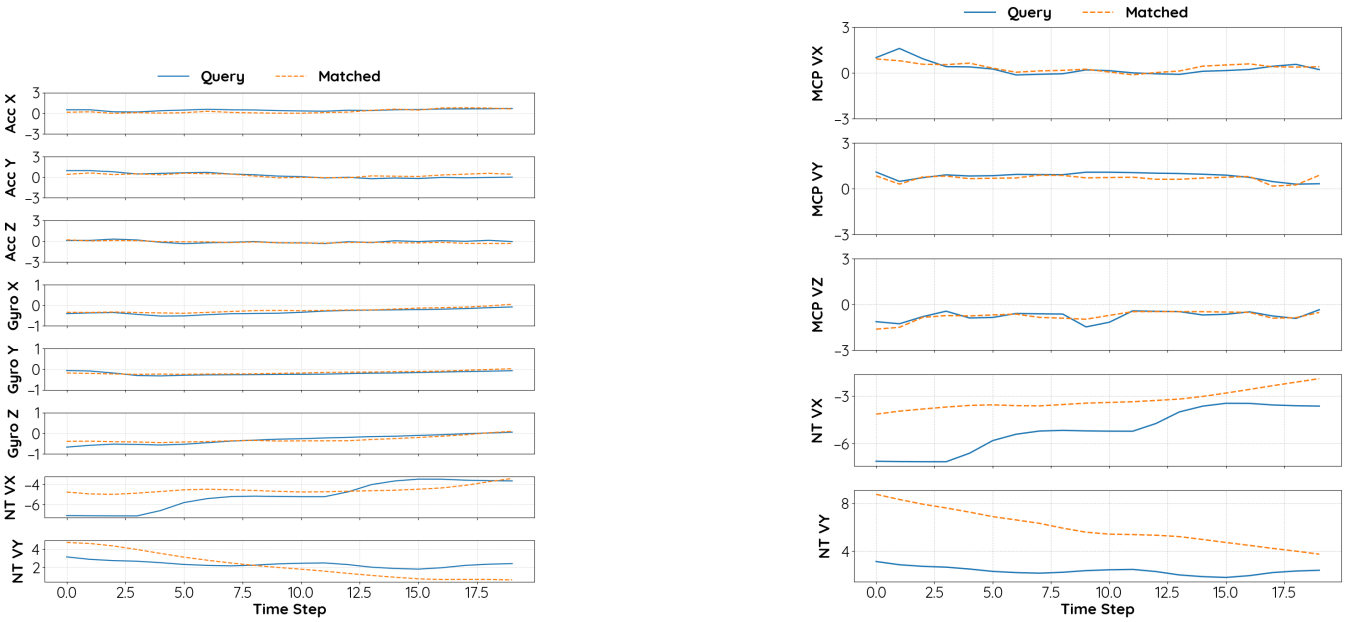
[63] Jianfei Zhang and Yongqiang Shi. 2024. A personalized federated learning method based on clustering and knowledge distillation. *Electronics* 13, 5 (2024), 857.

[64] Xinyong Zhang. 2021. Evaluating the effects of saccade types and directions on eye pointing tasks. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 1221–1234.

[65] Hao Zhou, Taiting Lu, Yilin Liu, Shijia Zhang, and Mahanth Gowda. 2022. Learning on the rings: Self-supervised 3d finger motion tracking using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–31.

[66] Yexu Zhou, Haibin Zhao, Yiran Huang, Till Riedel, Michael Hefenbrock, and Michael Beigl. 2022. TinyHAR: A Lightweight Deep Learning Model Designed for Human Activity Recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers* (Cambridge, United Kingdom) *(ISWC '22)*. Association for Computing Machinery, New York, NY, USA, 89–93. doi:10.1145/3544794.3558467

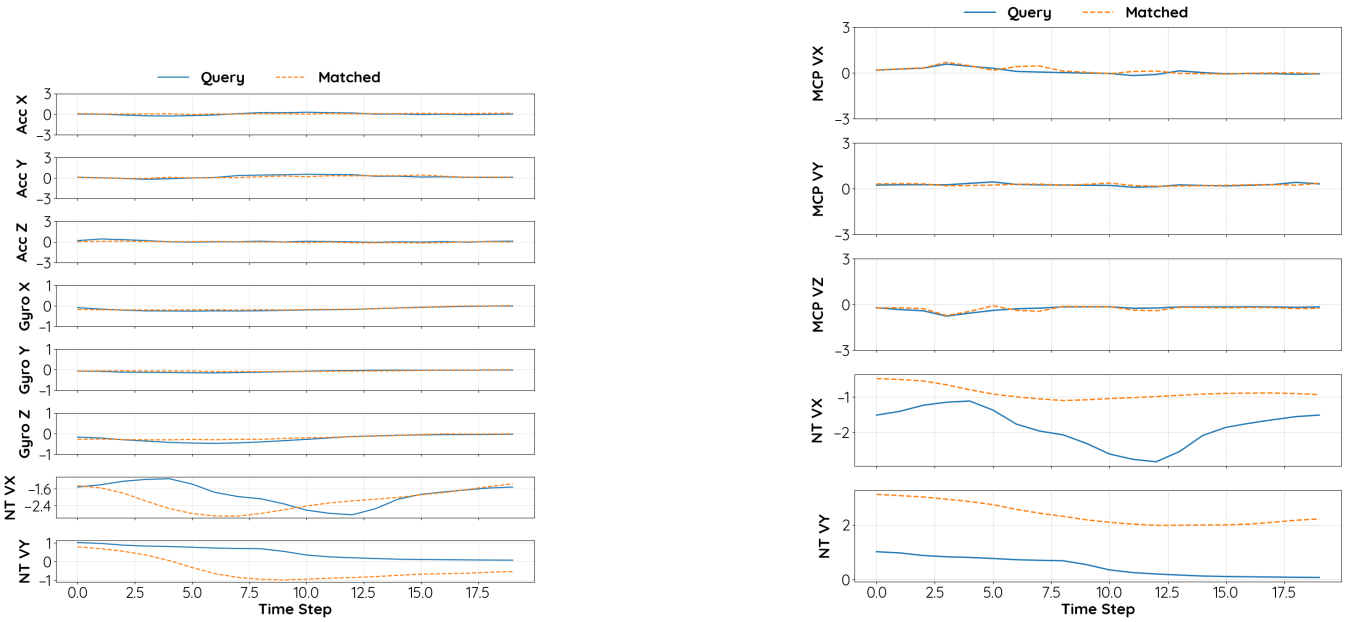
A More Examples for the Mapping from MCP to Nail Tip



(a) The closest IMU segment comparison.

(b) The closest optical motion capture segment comparison.

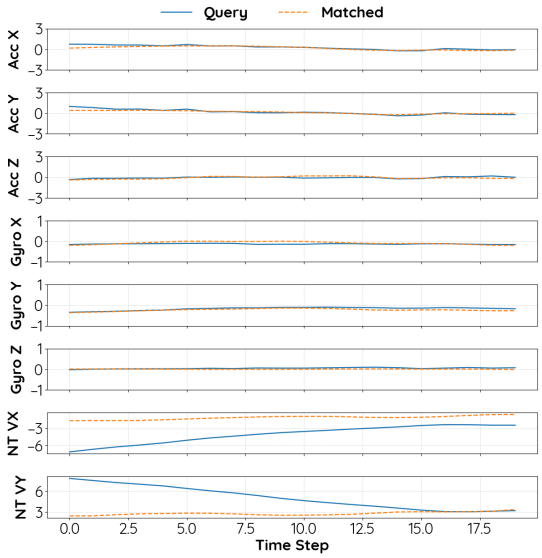
Figure 16: More examples to show the data segment at MCP as the query with the matched segment close in distance. Even if the data at MCP appear almost identical, the motion data of the fingertips show significant differences.



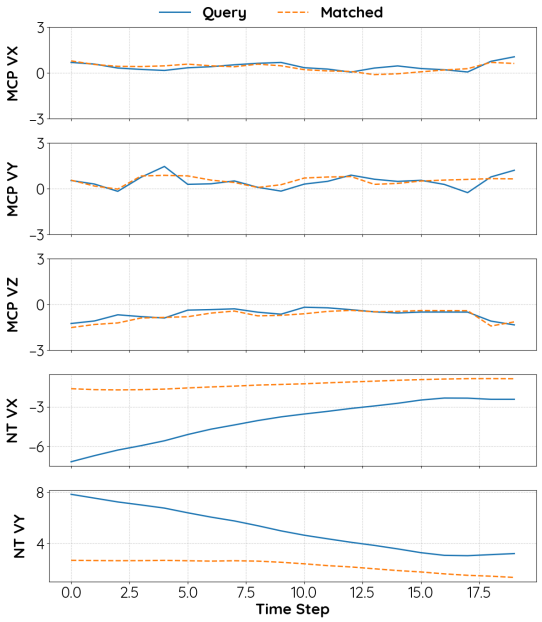
(a) The closest IMU segment comparison.

(b) The closest optical motion capture segment comparison.

Figure 17: More examples to show the data segment at MCP as the query with the matched segment close in distance. Even if the data at MCP appear almost identical, the motion data of the fingertips show significant differences.



(a) The closest IMU segment comparison.



(b) The closest optical motion capture segment comparison.

Figure 18: More examples to show the data segment at MCP as the query with the matched segment close in distance. Even if the data at MCP appear almost identical, the motion data of the fingertips show significant differences.